On Expected Probabilistic Polynomial-Time Adversaries: A Suggestion for Restricted Definitions and Their Benefits*

Oded Goldreich
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded.goldreich@weizmann.ac.il

July 13, 2009

Abstract

This paper concerns the possibility of developing a coherent theory of security when feasibility is associated with *expected* probabilistic polynomial-time (*expected* PPT). The source of difficulty is that the known definitions of *expected* PPT *strategies* (i.e., *expected* PPT *interactive* machines) do not support natural results of the type presented below.

To overcome this difficulty, we suggest new definitions of *expected* PPT strategies, which are more restrictive than the known definitions (but nevertheless extend the notion of *expected* PPT *non-interactive* algorithms). We advocate the conceptual adequacy of these definitions, and point out their technical advantages. Specifically, identifying a natural subclass of black-box simulators, called *normal*, we prove the following two results:

1. Security proofs that refer to all *strict* PPT adversaries (and are proven via normal black-box simulators) extend to provide security with respect to all adversaries that satisfy the restricted definitions of *expected* PPT.
2. Security composition theorems of the type known for *strict* PPT hold for these restricted definitions of *expected* PPT, where security means simulation by normal black-box simulators.

Specifically, a normal black-box simulator is required to make an expected polynomial number of steps, when given oracle access to *any strategy, where each oracle call is counted as a single step.* This natural property is satisfies by most known simulators and is easy to verify.

Keywords: Zero-Knowledge, Secure Multi-Party Computation, Protocol Composition, Black-Box Simulation, Reset Attacks, Expected Probabilistic Polynomial-Time.

*An extended abstract has appeared in the proceedings of the *4th Theory of Cryptography Conference*, Springer LNCS, Vol. 4392, pages 174–193, 2007. This research was partially supported by the Israel Science Foundation (grant No. 460/05).

Contents

1 An Opinionated Introduction

The title of this introduction and the use of first person singular in its text are meant to indicate that this introduction is more opinionated than is customary in our field. Nevertheless, I will try to distinguish facts from my opinions by using adequate phrases.

In my opinion, the first question that should be asked when suggesting and/or reviewing a definition is what is the purpose of the definition. When reviewing an existing definition, a good way to start is to look into the history of the definition, since the purpose may be more transparent in the initial works than in follow-up ones.

Before turning to the history and beyond, let me state that I assume that the reader is familiar with the notion of zero-knowledge and the underlying simulation paradigm (see, e.g., [G01, Sec. 4.3.1]). In fact, some familiarity with general secure multi-party computation (e.g., at the overview level of [G04, Sec. 7.1]) is also useful. Indeed, this paper is not intended for the novice: it deals with subtle issues that the novice may (or even should) ignore.

1.1 The history of related definitions

To the best of my recall, the first appearance in cryptography of the notion of expected (rather than strict) probabilistic polynomial-time was in the seminal work of Goldwasser, Micali, and Rackoff [GMR]. The reason was that the simulators presented in that paper (for the Quadratic Residuosity and the Quadratic Non-Residuosity interactive proofs) were only shown to run in *expected* probabilistic polynomial-time.[1] Recall that these simulators were used in order to simulate the interaction of arbitrary *strict* probabilistic polynomial-time (adversarial) verifiers with the honest prover.

At the time, the discrepancy between the expected probabilistic polynomial-time allowed to the simulator and the restriction of the adversary to strict probabilistic polynomial-time did not bother anybody. One reason for this lack of concern seems to be that everybody was overwhelmed by the new fascinating notion of zero-knowledge proofs, its mere feasibility, and its wide applicability (as demonstrated by [GMR, GMW]). But as time passed, some researchers became bothered by this discrepancy, which seemed to violate (at least to some extent) the intuition underlying the definition of zero-knowledge. Specifically, relating the complexity of the simulation to the complexity of the adversary is the essence of the simulation paradigm and the key to the conclusion that the adversary gains noting by the interaction (since it can obtain the same, essentially as easily, without any interaction). But *may we consider expected polynomial-time and strict (probabilistic) polynomial-time as being the same complexity*?

The original feeling was that the discrepancy between strict and expected polynomial-time is not very significant, and I do hold this view to this very day. It is telling that everybody seems quite happy with replacing one polynomial (bound of the running time) by another, at least as a very first approximation of the intuitive notion of similar complexity.[2] Still, I cannot deny that

[1]Note that while a small definitional variation (cf. [G01, Sec. 4.3.1.1] versus [G01, Sec. 4.3.1.6]) suffices for obtaining a strict probabilistic polynomial-time (perfect) simulation for the QR protocol, this does not seem to be the case when the QNR protocol is concerned. The same dichotomy is manifested between the Graph Isomorphism and Graph 3-Colorability protocols (of [GMW]) on the one hand and the constant-round zero-knowledge proof of [GK96] on the other hand. The dichotomy arises from two different simulation techniques; the first is tailored for "challenge-response" protocols, while the second refers to the use of "proofs-of-knowledge" (which may be implicit and trivial (as in [GK96])). Indeed, recall that the definition of proofs-of-knowledge refers explicitly to expected running-time (cf., e.g., [G01, Sec. 4.7.1]).

[2]Indeed, my advocacy of *knowledge tightness* [G01, Sec. 4.4.4.2], a notion aimed at quantitatively bounding the ratio of the running times of the simulator and adversary, has never gain much attention. (And yes, I am aware of

there is something unpleasing about this discrepancy. Following [KL08], let me refer to this issue as an aesthetic consideration.

Jumping ahead in time, let me mention a more acute consideration articulated in [KL08]: A different handling of adversaries and simulations (e.g., the discrepancy between expected polynomial-time and strict probabilistic polynomial-time) raises technical difficulties and, in particular, stands in the way of various desired composition theorems (e.g., of the type presented in [GO94, C00]). But let me get back to the story.

Faced with the aforementioned aesthetic consideration, a few researchers suggested a simple solution: extending the treatment of adversaries to ones running in expected polynomial-time. This suggestion raised a few problems, the first being *how to define expected polynomial-time interactive machines*? (In addition, there are other problems, which I will discuss later.)

Feige's proposal [F90] was to consider the running-time of the adversary when it interacts with the honest party that it attacks, and require that the adversary runs in expected polynomial-time (in such a random interaction). My own proposal was to allow only adversaries that run in expected polynomial-time regardless with whom they interact; that is, the adversary is required to run in expected polynomial-time when interacting with any other strategy. Feige objected to my proposal saying that it unduly restricts the adversary, which is designed to attack a specific strategy and thus should be efficient only when attacking this strategy. My own feeling was that it is far more important to maintain a coherent theory by using a "stand-alone" notion of expected polynomial-time; that is, a notion that categorizes strategies regardless of their aim (e.g., without reference to whether or not these strategies model adversaries (and which strategies these adversaries attack)). The rationale underlying this feeling is discussed in Section 1.2. (Another objection to Feige's definition refers to the fact that, when applying its underlying principle to the standard definition of *strict* probabilistic polynomial-time, the result is a significantly larger class of adversaries, which includes adversaries that may not even halt when interacting with strategies other than those they were designed to attack.)

In any case, a major problem regarding the suggestion of extending the treatment of adversaries to ones running in expected polynomial-time is *whether such an extension is at all possible.* One specific key question is *whether known simulators can handle expected polynomial-time adversaries.* As pointed out in [KL08], in some cases (e.g., the simulator of [GK96]), the answer is negative even if one uses the more restricted notion of expected polynomial-time adversaries (which refers to interaction with any possible strategy). Another important question is *whether composition theorems that are known to hold for strict probabilistic polynomial-time* (strategies and simulators) *can be extended to the case of expected polynomial-time* (strategies and simulators).

Indeed, the "question of composition" became a major concern in the 1990's and motivated a re-examination of many aspects of the theory of cryptography. Here I refer specifically to the Sequential Composition Theorem of Canetti [C00], which supports modular construction of protocols, and to the Concurrent Composition Theorem of Canetti [C01], which is aimed at preserving security in settings where numerous executions of arbitrary protocols are taking place concurrently. These composition results were obtained when modeling adversaries as *strict* probabilistic polynomial-time strategies and allowing only *strict* probabilistic polynomial-time simulators. One consequence of the lack of analogous results for the case of *expected* polynomial-time was that the modular construction of secure protocol had to avoid protocols that were only known to be simulateable in *expected* polynomial-time.[3]

---

the recent work of Micali and Pass [MP06] that introduces and advocates an even more refined notion.)

[3]For example, relatively efficient proofs-of-knowledge (which only guarantee *expected* polynomial-time extraction) were avoided (e.g., in [G04, Sec. 7.4.1.3]) and strong proofs-of-knowledge (cf. [G01, Sec. 4.7.6]) were used instead.

Recently, Katz and Lindell [KL08] initiated a study of the possibility of simulating *expected* polynomial-time adversaries and/or obtaining composition theorems (or sufficiently good alternatives) for the case of *expected* polynomial-time. They showed that in some cases (e.g., when the simulator satisfies some additional properties and/or under some super-polynomial intractability assumptions) such partial results can be obtained.[4] These results do not provide a "free" transformation from the *strict* probabilistic polynomial-time model to the *expected* polynomial-time model, where "free" means without referring to additional assumptions. In my opinion, as long as this is the state of affairs, one better look for alternative directions.

1.2 Towards new definitions

My starting point (or thesis) is that *we should not care about expected polynomial-time adversaries per se*. As hinted by my historical account, researchers were perfectly happy with strict probabilistic polynomial-time adversaries and would have probably remained so if it were not for the introduction of expected polynomial-time simulators. Indeed, at the end of the day, the user (especially a non-sophisticated one) should care about what an adversary can obtain within a specific time (or various possible amounts of work), where the term 'obtain' incorporates also a quantification of the success probability. I claim that our goal as researchers is to provide such statements (or rather techniques for providing such statements), and that *expected polynomial-time machines may appear in the analysis only as intermediate steps* (or mental experiments).

My thesis is further enforced by the confusing and nonintuitive nature of expected running-time, especially when applied in the context of cryptography, and by numerous annoying phenomena related to expected-time complexity.

The cryptographic angle. A typical security analysis refers both to the running-time and the success probability of a possible attack. Usually the former is fixed (i.e., to strict polynomial-time), and so one may discuss the latter separately. However, in my opinion, when the running-time is a random variable, providing only the expected running-time and the overall success probability is quite meaningless, because the success is likely to be correlated with the running-time. Instead, one should keep track of both the running-time and the success probability; that is, provide an estimate of the success probability per each approximate value of the running-time (i.e., assert that with probability $p(t)$ the attack runs for t steps, and condition on this event it succeeds with probability $s(t)$).

The generic phenomena. The point is that, unlike strict polynomial-time, expected polynomial-time is a *highly non-robust notion that is not preserved under changes of computational model and standard algorithmic compositions*. These "features" are an artifact of the "bad interaction" between the expectation operator and many non-linear operators: for example, for a random variable X, we cannot upper-bound $\mathrm{E}[X^2]$ as a function of $\mathrm{E}[X]$. Thus, if X is a random variable that represents the running-time of some process Π (where the probability space

[4] Roughly speaking, one of their results provides a transformation of some simulators that handle strict probabilistic polynomial-time adversaries into simulators that handle expected polynomial-time adversaries, while assuming that the original simulator's queries are strongly indistinguishable from the messages of the real protocol. Another result provides a composition theorem for expected polynomial-time simulators (which handle strict probabilistic polynomial-time adversaries), while relying on strongly pseudorandom functions. In both cases, the term *strong* refers to versions of computational indistinguishability that are required to hold with respect to super-polynomial-time observers. This means that for obtaining (ordinary) computational security, somewhere along the way, one needs to make a super-polynomial-time intractability assumption. Also note that the simulators constructed in [KL08] use the corresponding adversaries in a "slightly non-black-box" manner in the sense that they terminate executions (of these adversaries) that exceed a specific number of steps.

is that of the internal coin tosses of Π), then we cannot bound the expected running-time of various modest variants of Π (e.g., which square its running-time) in terms of the expected running-time of Π. (See Footnote 26, which refers to a natural case in which this problem arises, and [G97] for an analogous discussion of the effect of this problem in the context of average-case complexity.)

The foregoing reservations regarding expected polynomial-time are of lesser concern when expected running-time is only used as an intermediate step (rather than as a final statement). Taking this approach to its extreme, I claim that for this purpose (of an intermediate step) it is legitimate to use any (reasonable) definition of expected polynomial-time strategies, and that among such possibilities we better select a definition that supports the desired results (e.g., simulation of corresponding adversaries and composition theorems). Thus, we should seek a definition of expected polynomial-time strategies that enjoys the following properties:

1. The definition should include all strict probabilistic polynomial-time strategies (but should not extend "much beyond that"; e.g., super-polynomial-time computations may only occur with negligible probability).

2. When applied to non-interactive strategies (i.e., stand-alone algorithms) the definition of expected polynomial-time strategies should yield the standard notion of expected polynomial-time.

   This property is not only a matter of aesthetic considerations but is rather important for composition theorems (as desired in Property 3b). Furthermore, when applied to the context of zero-knowledge, the current property implies that expected polynomial-time simulators are deemed admissible by this definition.[5]

3. The definition should allow to derive the results that we seek:

   (a) Known simulators that handle strict probabilistic polynomial-time adversaries should also handle adversaries that satisfy the definition.[6]

   (b) The definition should support natural composition theorems (e.g., of the type proven by Canetti [C00]).

With the foregoing properties in mind, let me suggest a couple of new definitions of expected polynomial-time strategies. These definitions will be more restrictive than the existing definitions of this notion (which were reviewed in Section 1.1).

1.3 The new definitions

Looking at the problem of simulating an "expected polynomial-time" adversary (cf. [KL08]), it becomes evident that the source of trouble is the fact that the bound on the running-time of the adversary (w.r.t any real interaction) is no longer guaranteed when the adversary is invoked by a simulator. The point is that the queries made by the simulator may have a different distribution than the messages sent in any real interaction (especially, since some of these queries may not

[5]In fact, we should strengthen Property 2 by requiring that also in the context of secure multi-party computation (where the simulators are themselves interactive machines) the known "expected polynomial-time" simulators (of strict probabilistic polynomial-time) are deemed admissible by the selected definition.

[6]Actually, we may relax this condition by allowing a modification of the simulator but not of the protocol and/or the underlying intractability assumptions.

appear in the transcript output by the simulator). Furthermore, the simulator is resetting the adversary, which may allow it to find queries that are correlated to the adversary's internal coin tosses in ways that are unlikely to happen in any real interaction (see examples in [KL08] and in the proof of Proposition 5). Such queries may cause the adversary to run for a number of steps that is not polynomial on the average. Indeed, this problem does not occur in the case of strict probabilistic polynomial-time adversaries because in that case we have an *absolute bound* on the number of steps taken by the adversary, regardless of which messages it receives.

Let me stress that assuming that the adversary runs in expected polynomial-time when interacting with any other party does not solve the problem, because the distribution of the simulator's queries may not correspond to the distribution of an interaction with any standard interactive machine. The simulator's queries correspond to a "reset attack" on the adversary, where reset attack are as defined in [CGGM] (except that here they are applied on the adversary's strategy rather than on the honest party's strategy). Specifically, in a reset attack, the internal coin tosses of the strategy are fixed (to a random value) and the attacker may interact several times with the resulting residual (deterministic) strategy.

The foregoing discussion suggests a simple fix to the problem. Just define expected polynomial-time strategies as ones that run in expected polynomial-time under any reset attack that interact with them for a polynomial number of times. Actually, we should allow attacks that interact with these strategies for an expected polynomial number of times.[7] (See Definition 3.)

It seems that any (black-box) simulator that handles strict probabilistic polynomial-time adversaries can also handle adversaries that run in expected polynomial-time under the foregoing definition. After all, this definition was designed to support such a result. However, I was not able to prove this result without further restricting the class of simulators (in a natural way). For details, see Section 1.4.

But before turning to the results, let me suggest an even more restricted notion of expected polynomial-time strategies. I suggest to consider strategies that run in expected polynomial-time when interacting with any ("magical") machine that receives the strategy's internal coin tosses as side information. Arguably, this is the most restricted (natural) notion of expected polynomial-time strategies (which, when applied to non-interactive machines, coincides with the standard definition of expected polynomial-time). Needless to say, this definition (which is more restrictive than the aforementioned resetting definition) also supports the extension of simulators that handle strict probabilistic polynomial-time adversaries to handle adversaries satisfying the current definition.

Clearly, both definitions satisfy the first two desirable properties stated in Section 1.2. As for the third desirable property, it will be at the focus of the next subsection.

1.4 The main results

In a nutshell, the main results establish the third desirable property for both the (new) definitions, when assuming that the provided simulators (i.e., the simulators provided by the corresponding hypothesis) belong to a natural subclass of black-box simulators. Indeed, one could hope that these results would hold for all (universal) simulators or at least for all black-box simulators.[8]

The issue at hand is the definition of efficient black-box simulators. Since black-box simulators are typically given oracle access to an efficient strategy, some texts only refer to what happens

[7] When measuring the expected number of interactions, I refer to a variant of Feige's notion of expected complexity with respect to the designated machine. Indeed, this widens the class of possible (reset) attackers, which further limits the class of admissible strategies (i.e., those that are expected polynomial-time under such attackers).

[8] Recall that a universal simulator is a universal machine that is given (as input) the code of the adversary that it simulates. In contrast, a black-box simulator is only given oracle access to the corresponding strategy.

in such a case (and mandate that the overall simulation be efficient, where one also accounts for the steps of the strategy). A more natural and robust definition mandates that the number of steps performed by the black-box simulator itself be feasible, when the simulator is given oracle access to *any strategy*. Specifically, I consider black-box simulators that make an *expected* number of steps that is upper-bounded by a fixed polynomial in the length of the input, *where each oracle call is counted as a single step*, and call such a simulator normal.[9] Indeed, the known (black-box) simulations including those that run in *expected* polynomial-time (e.g., [GK96]) are normal. Furthermore, in my opinion, the notion of a normal black-box simulator fits the natural formulation of the notion of an expected probabilistic polynomial-time oracle machine, because it is natural to require that the complexity of an oracle machine (unlike its output) be independent of the oracle that it accesses. (For further discussion see the beginning of Section 3 (including Footnote 23).)

Turing back to the main results, recall that the new definitions (or actually the "resetting-based" one) were devised to support the first main result (stated in Theorem 10). This result asserts that *any normal black-box simulator that handles strict probabilistic polynomial-time adversaries can also handle adversaries that run in expected polynomial-time under the new definition(s).* In particular, it implies that normal black-box zero-knowledge protocols remain simulateable when attacked by adversaries that satisfy the new definition(s) of expected polynomial-time. This applies, in particular, to the proof system of [GK96], for which analogous ("free") results were not known under the previous definitions of expected polynomial-time.[10]

Note that the fact that the aforementioned (normal black-box) simulations run in *expected* polynomial-time also when given access to any *expected* polynomial-time adversary is quite obvious from the new definition(s). This follows from the fact that normal black-box simulators invoke the adversary strategy for an *expected* polynomial number of times, while the "resetting-based definition" upper-bounds the total *expected* time consumed by the adversary in such invocations. What should be shown is that, also in this case, the corresponding simulation produces good output (i.e., indistinguishable from the real interaction). This can be shown by using a rather straightforward "truncation" argument.[11]

Let us now turn to the question of composition, starting with the sequential composition of zero-knowledge protocols. The known result (of [GO94]) refers to *strict* probabilistic polynomial-time adversaries (and holds both with respect to strict and expected polynomial-time simulation).[12] However, the known argument does not extend to *expected* polynomial-time adversaries. Recall that the said argument transforms any adversary that attacks the composed protocol into a residual adversary that attacks the basic protocol. The source of trouble is that the fact that the former adversary is expected polynomial-time (under any definition) does not imply that the latter adver-

[9] In contrast, the number of steps made by an oracle machine that is not normal may not be "uniformly bounded" over all possible oracles.

[10] As in Section 1.1, by free results we mean positive results that do not rely on additional assumptions. Recall that Katz and Lindell [KL08] showed that the simulator presented in [GK96] fails (w.r.t expected polynomial-time under the previous definitions). Their work implies that, if strongly hiding commitment schemes are used in the protocol, then an alternative simulator does work. In contrast, my result applies to the simulator presented in [GK96] and does not require strengthening the commitment scheme used in the protocol. Furthermore, the running-time is preserved also for no-instances (cf., in contrast, [KL08, Sec. 3.3]).

[11] Indeed, the running-time analysis relies on the hypothesis that the simulator is normal, whereas the analysis of its output only relies on the hypothesis that the simulator is black-box. In contrast, for the claim (of Theorem 10) itself to make sense at all it suffices to have a universal simulator. (Note that in the (hard to conceive) context of "non-universal" simulation it is not clear what we mean by saying that a simulator that handles any $A \in \mathcal{C}$ can handle any $A' \in \mathcal{C}'$.)

[12] The original proof (of [GO94]) refers to strict polynomial-time simulators, but it extends easily to expected polynomial-time simulators.

sary is expected polynomial-time (under this definition). See the proof of Theorem 9 for details. Fortunately, there is an alternative way: just note that the simulator obtained by [GO94], which refers to *strict* probabilistic polynomial-time adversaries, can handle *expected* polynomial-time adversaries (i.e., by invoking Theorem 10 (or rather its zero-knowledge version – Theorem 8)).

The foregoing idea can also be applied to the general setting of secure multi-party computation, but additional care is needed to deal with the extra complexities of this setting (as described next). Specifically, the so-called *sequential composition theorem* of Canetti [C00] (see also [G04, Sec. 7.4.2]) refers to an oracle-aided (or "hybrid") protocol Π that uses oracle calls to a functionality[13] f, which can be securely computed by a protocol ρ. (Note that the corresponding oracle-aided protocol was not mentioned in the context of zero-knowledge, because it is trivial (i.e., it merely invokes the basic protocol several times).) The theorem asserts that the security of Π (with respect to a specific functionality unmentioned here) is preserved when Π uses subroutine calls to ρ rather than oracle calls to f. *This result refers to security with respect to strict probabilistic polynomial-time adversaries that is demonstrated by strict probabilistic polynomial-time simulators.* One point to notice is that the proof of security of the resulting protocol, denoted Π', proceeds by incorporating the simulator of ρ into an adversary for Π. Thus, if the simulator of ρ runs in *expected* polynomial-time then so does the resulting adversary (for Π), and thus the simulator for Π has to handle *expected* polynomial-time adversaries (even if we only care of *strict* polynomial-time adversaries attacking Π'). Indeed, having a simulator for Π that handles any *expected* polynomial-time adversaries suffices for a partial result that refers to *strict* probabilistic polynomial-time adversaries for the resulting protocol Π' and to *expected* polynomial-time simulators (for ρ, Π, and Π'). The general (sequential) composition theorem for the case of expected polynomial-time (which refers to expected polynomial-time adversaries and simulators) follows by applying Theorem 10.

An important corollary to the foregoing extendability and composition theorems (i.e., Theorem 10 and 11) asserts that it is possible to compose secure protocols, *when security is demonstrated via expected polynomial-time simulators but refers only to strict probabilistic polynomial-time adversaries.* In such a case, the extendability theorem allows to use these simulators with respect to *expected* polynomial-time adversaries, whereas the composition theorem applies to the latter. Thus, one may freely use *expected* polynomial-time simulators, and be assured that the corresponding secure protocols can be composed (just as in the case that their security is demonstrated via *strict* polynomial-time simulators).

Turning to the concurrent composition theorem of Canetti [C01], recall that it evolves around the notion of environmental security (a.k.a UC-security [C01]). Specifically, Canetti proved that any protocol that is environmentally secure preserves security under arbitrary concurrent executions, where the adversaries, simulators, and environments are all modeled as *strict* probabilistic polynomial-time strategies (with non-uniform auxiliary inputs for the environments). He then suggested the methodology of establishing environmental-security as a way of obtaining security under concurrent composition. Consequently, an extension of Canetti's methodology to the *expected* polynomial-time setting requires (1) verifying that Canetti's proof extends to this setting, and (2) obtaining environmental security for *expected* polynomial-time adversaries and environments. Using the new definitions of expected polynomial-time strategies, the first requirement follows analogously to the proof of the sequential composition theorem, while the second requirement follows by generalizing Theorem 10 (which may be viewed as referring to trivial environments).

The bottom-line is that, for normal black-box simulators, the new definitions of expected polynomial-time strategies provide a "free" transformation from the *strict* probabilistic polynomial-time model to the *expected* polynomial-time model. In particular, *normal black-box simulators that*

[13] A functionality is a randomized version of a multi-input multi-output function (cf. [G04, Sec. 7.2.1]).

work in the strict model extend to the expected model, and the most famous composition theorems extend similarly.

1.5 Why deal with expected polynomial-time at all?

In light of the difficulties discussed in Section 1.1, one may ask *why do we need this headache* (of dealing with expected polynomial-time) *at all*? This question is further motivated by my views (expressed in Section 1.2) by which *we should not care about expected polynomial-time adversaries per se*. The answer, as hinted in Section 1.1, is that *we do care about expected polynomial-time simulators*.

Specifically, some natural protocols are known to be secure (or zero-knowledge) only when the definition of security allows expected polynomial-time simulators. A notable example, already mentioned several times, is the constant-round zero-knowledge proof system of [GK96]. Furthermore, as proved in [BL02], constant-round proof system for sets outside $\mathcal{BPP}$ do not have strict polynomial-time black-box simulators (although they do have such non-black-box simulators [B01], which are less preferable for reasons discussed below).

In general, expected polynomial-time simulators seem to allow more efficient protocols and/or tighter security analysis. Whereas various notions of protocol efficiency are well-understood, a few words about the tightness of various security analyses are in place. Loosely speaking, *security tightness*[14] is essentially the ratio between the running-time of the adversary and the (expected) running-time of the simulator that handles it: The security tightness of a protocol is a lower-bound on the aforementioned ratio that essentially holds for every probabilistic polynomial-time adversary; that is, a protocol is said to have security tightness τ if there exists a polynomial q_0 such that, for every polynomial p, every p-time adversary is simulated within time $\tau^{-1}p + q_0 + q_0$. Indeed, in many cases (also when strict polynomial-time simulators exist), the expected running-time of the simulator provides a better bound than the worst-case running-time of the simulator.

In my opinion, security tightness should serve as a major consideration in the evaluation of alternative protocols, and claims about protocol efficiency are almost meaningless without referring to their security tightness. For example, in many cases, modest parallelization can be achieved at the cost of a deterioration in the security tightness (cf. [G01, Sec. 4.4.4.2]). Let me stress that, by definition, black-box simulators always yield a *noticeable* bound on the security tightness (and in some cases they offer a constant bound), whereas non-black-box simulators may fail to have such bound (e.g., indeed, that's the case with Barak's simulators [B01]).[15]

Thus, I suggest the following methodology: When designing your protocol and proving its security, allow yourself expected polynomial-time simulations. To assist the design and analysis, use the "extendability results" (e.g., Theorem 10) provided in this work as well as relevant composition theorems (e.g., Theorem 11). Finally, when obtaining the desired protocol with a security analysis that refers to an expected polynomial-time simulator, you may interpret it as providing a trade-off between the simulation time and the corresponding deviation (from the real interaction). But actually, a final claim that refers to expected simulation time may be as appealing when stated in terms of security tightness (e.g., the effect of any strict polynomial-time adversary can be achieved by a simulation that is expected to run three times as long).

[14] In the special case of zero-knowledge, the corresponding notion is called *knowledge tightness* [G01, Sec. 4.4.4.2]. Note a minor technicality: here tightness is define as the reciprocal of the ratio in [G01, Sec. 4.4.4.2].

[15] As usual, a noticeable function is one that decreases slower than the reciprocal of some positive polynomial. Thus, noticeable security tightness means that there exists a polynomial q such that, for every polynomial p, every p-time adversary is simulated within time $q \cdot p$. But if the simulation of p-time adversaries requires time p^3, then the protocol does not have a noticeable security tightness.

Indeed, my opinion is that *there is no contradiction between not caring about expected polynomial-time adversaries and providing security guarantees that refer to the expected simulation time*: Whereas (at least potentially) the adversary is a real entity, its simulation is (always) a mental experiment. Furthermore, I believe that the foregoing methodology may yield the best trade-offs between the efficiency of the protocol and the tightness of its security.

Finally, let me note that there are alternative ways of handling the problems that motivate the introduction of expected polynomial-time to Cryptography (i.e., the failure of strict polynomial-time simulation in some cases). These alternatives are based on different notions of "typical efficiency" that are applicable to "varying" running-time (i.e., running-time that is expressed as a random variable). In each case, one should start with a definition that refers to standard algorithms, and extend it to a definition that refers to interactive machines. For details, see Section 5. Indeed, the issues arising in such extensions are the same as the ones discussed throughout the rest of this paper. It is my belief, however, that expected running-time (as treated in the rest of this paper) provides the best trade-offs between the efficiency of the protocol and the tightness of its security.

1.6 On the treatment of concurrent composition

As stated at the end of Section 1.4, the new definitions of expected probabilistic polynomial-time strategies allow to extend the known sequential and concurrent composition theorems from the strict PPT setting to the expected PPT setting. However, in my opinion, there is a significant difference between the importance of these extensions (which correspond to these two cases). The difference is rooted in the difference between the original theorems (i.e., their strict PPT versions).

Recall that the sequential composition theorem holds for every secure protocol (i.e., "stand-alone security" suffices [C00]), whereas concurrent composition essentially requires a stronger notion of security (i.e., environmental security [C01, L09]). This difference in the security level seems to be reflected by a difference in the variety of proof techniques (i.e., simulation techniques), where more stringent security leaves room for less techniques. Indeed, while stand-alone security is sometimes demonstrated by using expected PPT simulators, all known demonstrations of environmental security employ strict PPT simulators. Furthermore, my feeling is that there is no benefit in using expected PPT simulators towards demonstrating environmental security (with respect to strict PPT adversaries). Thus, unless my feeling is wrong, I see no real motivation for extending the concurrent composition theorem to the expected PPT setting. Nevertheless, I provided such an extension in order to show that the new definitions of expected PPT strategies are not incompatible with such an extension.

1.7 Organization

Section 2 provides formal statements of the aforementioned (old and new) definitions as well as a demonstration of a hierarchy among them. Since the special case of zero-knowledge protocols provides a good benchmark for the general case of secure protocols, the main results are first presented in that setting (see Section 3). This simplifies things, because in that special case the simulators are standard algorithms rather than interactive strategies (for the so-called "ideal-model"; see, e.g., [G04, Sec. 7.2]). Nevertheless, I believe that the main ideas are already present in the zero-knowledge setting, and that this belief is supported by the treatment of general protocols (provided in Section 4). Section 5 demonstrates the applicability of the main approach to alternatives measures of "varying" running-time. Section 6 contains conclusions and open problems.

The exposition in Sections 4.3 and 5 is significantly less detailed and more sketchy than the rest of the paper. Indeed, these sections should be viewed as a demonstration of the feasibility of applying the main definitional approach also to these settings. My choice to provide only a sketchy exposition of these applications is related to my reservations regarding their importance (see Section 1.6 and the end of Section 1.5, respectively).

The (security) tightness lens. Paragraphs with this heading are meant to facilitate the methodology suggested in Section 1.5, but they may be ignored at first reading.

2 The Definitions

We adopt the standard terminology of interactive machines, while occasionally identifying strategies (which specify the next message to be sent by an interactive machine given its view so far) with the interactive machines that activate them. We use the shorthand PPT for *probabilistic polynomial-time* whenever using the full term is too cumbersome; typically, we do so when contrasting strict PPT and expected PPT. For simplicity, we only consider the two-party case. We denote by x the common (part of the) input, and denote by y and z the corresponding private inputs of the two parties. The reader may ignore y and z, which model (possibly non-uniform) auxiliary information.

Additional (standard) conventions. We state the complexity of interactive machines (and strategies) as a function of the length of the common input, x; consequently, some time-bounded machines cannot read their entire private inputs (i.e., y or z, resp.). For sake of brevity, we sometimes say that some quantity (e.g., number of steps) is polynomial (or exponential), rather than saying that it is polynomial (or exponential) in the length common input, but this is always the intention. In the actual technical treatment we assume that all computations of all machines halt in finite time, and furthermore that this time is bounded by a single function in the length of the common input.[16] (Note that, in all reasonable cases, this restriction can be easily enforced by truncating all runs after an exponential number of steps.)

2.1 Known definitions

We start by formulating the two known definitions that were mentioned in Section 1.1.

Definition 1 (Feige [F90]): *The strategy σ is* expected PPT w.r.t a specific interactive machine M_0 *if, for some polynomial p and every x, y, z, the expected number of steps taken by $\sigma(x, z)$ during an interaction with $M_0(x, y)$ is upper-bounded by $p(|x|)$, where the expectation is taken over the internal coin tosses of both machines.*

We stress that σ may be expected PPT with respect to some interactive machines but not with respect to others.

Definition 2 (attributed to Goldreich, e.g., in [KL08]): *The strategy σ is* expected PPT w.r.t any interactive machine *if, for some polynomial p, every interactive machine M, and every x, y, z, the expected number of steps taken by $\sigma(x, z)$ during an interaction with $M(x, y)$ is upper-bounded by $p(|x|)$.*

Here we may assume, without loss of generality, that M (which is computationally unbounded) is deterministic, and thus the expectation is only taken over the internal coin tosses of σ. The same convention is applied also in Definition 4 (but not in Definition 3; see discussion there).

[16] Hence, the probability space of all possible executions (on a fixed input) is finite, and so the expectation is always well-defined and finite.

2.2 New definitions

In the first new definition, we refer to the notion of a *reset attack* as put forward in [CGGM]. Such an attack proceeds as follows. First, we uniformly select and fix a sequence of internal coin tosses, denoted ω, for the attacked strategy σ, obtaining a residual deterministic strategy σ_ω. Next, we allow the attacker to interact with σ_ω numerous times (rather than a single time). Specifically, for each possible value of ω, the expected number of times that attacker interacts with σ_ω is upper-bounded by a fixed polynomial.[17]

Note that the attacker is not given ω explicitly, but its ability to (sequentially) interact with the residual strategy σ_ω for several times provides it with additional power (beyond interacting with σ itself for several times, *where in each interaction σ uses a fresh sequence of coin tosses*). As shown in [CGGM], such an attack is equivalent to a single interaction in which the attacker may (repeatedly) "rewind" σ (or rather σ_ω) to any prior point in the interaction and ask to resume the interaction from that point. Indeed, such an attack is reminiscent of the way that a (black-box) simulator uses an adversary strategy.

Definition 3 (tailored for simulation): *A q-*reset attack *on σ is an attack that, for every x, y, z and ω, interacts with σ_ω for an expected number of times that is upper-bounded by $q(|x|)$.[18] The strategy σ is* expected PPT w.r.t any reset attack *if, for some polynomial p, every polynomial q, every q-reset attack on σ, and every x, y, z, the expected total number of steps taken by $\sigma(x, z)$ during this attack is upper-bounded by $q(|x|) \cdot p(|x|)$.[19]*

We stress that the number of invocations of σ (like the total number of steps taken by σ) is a random variable defined over the probability space consisting of all possible interactions of the attacker and σ. Here (unlike in Definition 2), allowing the potential attacker to be probabilistic increases its power (and thus adds restrictions on strategies satisfying the definition). The reason is that, for each fixed ω, the number of times that a probabilistic attacker invokes σ_ω may be an arbitrary random variable with a polynomially bounded expectation (rather than being strictly bounded by a polynomial).

In the next (and last) definition, we consider a "magical" attacker that is given the outcome of the strategy's internal coin tosses as side information. That is, such an attack proceeds as follows. First, we uniformly select and fix a sequence of internal coin tosses, denoted ω, for the attacked strategy σ, obtaining a residual deterministic strategy σ_ω. Next, we provide the attacker with ω (as well as with z) and allow it a single interaction with σ_ω. We stress that this attacker is merely a mental experiment used for determining whether or not σ is expected polynomial-time (under the following definition).

[17] That is, there exists a polynomial p such that, for every ω, the expected number of times that the attacker interacts with σ_ω, on common input x, is at most $p(|x|)$. Note that we are upper-bounding the (expected) number of interactions initiated by the attacker (rather than its running-time). More importantly, the formulation of this restriction on the number of interactions is a hybrid of (the spirit of) Definitions 1 and 2: We are upper-bounding the (expected) number of interactions, not with respect to the designated σ, but rather with respect to each of the residual σ_ω. Finally, note that a simplified version that refers to the expected number of interactions with σ (i.e., the expectation is taken also over the coins of σ) yield a "bad" definition. (For example, suppose that σ_ω sends ω, and makes $2^{|\omega|}$ steps if $\omega = 1^{|\omega|}$ and halt immediately otherwise. Then, intuitively σ is expected PPT (and in fact it even satisfies Definition 4), but the reset attack that, upon receiving ω in the first interaction, invokes σ_ω for $2^{|\omega|}$ additional times if and only if $\omega = 1^{|\omega|}$, causes σ to make an expected exponential number of steps.)

[18] As in Definitions 1 and 2, such an attack is given x and y as its input.

[19] The upper-bounded of $q(|x|) \cdot p(|x|)$ seems natural; however, an upper-bounded of $p(|x| + q(|x|))$ would work just as well (for all results stated in this work), but would yield weaker quantitative bounds.

Definition 4 (seemingly most restrictive): *The strategy σ is* expected PPT w.r.t any magical machine *if, for some polynomial p, every interactive machine M' that is provided with the internal coin tosses of σ as side information, and every x, y, z, the expected number of steps taken by $\sigma(x, z)$ during an interaction with M' is upper-bounded by $p(|x|)$. That is, for a randomly selected ω, the expected number of steps taken by $\sigma_\omega(x, z)$ during its interaction with $M'(x, y, z, \omega)$ is upper-bounded by $p(|x|)$.*[20]

Here as in Definition 2, we may assume, without loss of generality, that M' (which is computationally unbounded) is deterministic, and thus the expectation is only taken over the internal coin tosses of σ. Thus, Definition 4 refers to the expectation, taken uniformly over all possible choices of ω, of the number of steps taken by (the residual deterministic strategy) $\sigma_\omega(x, z)$ during an interaction with (the deterministic strategy) $M'(x, y, z, \omega)$. Indeed, a strategy σ satisfies Definition 4 if and only if it *runs in expected polynomial-time even if each of the incoming messages is selected to maximize its running-time, when this selection may depend on the internal coin tosses of σ* (and its auxiliary-input z). This formulation is closest in spirit to the standard definition of strict PPT strategies.

2.3 Relating the definitions

It is easy to see that, for $i = 1, 2, 3$, Definition i+1 implies Definition i. In fact, it is not hard to see that the converses do not hold. That is:

Proposition 5 *For $i = 1, 2, 3$, the set of strategies that satisfy Definition i+1 is strictly contained in the set of the strategies that satisfy Definition i.*

Proof: The first two containments (i.e., for $i = 1, 2$) are plainly syntactic. Intuitively, the third containment (i.e., the fact that Definition 4 implies Definition 3) follows by noting that a reset attack does not add power to a computationally unbounded machine that gets σ's internal coin tosses. A rigorous proof of this fact follows.

Fixing an arbitrary q-reset attack A, denote by $T_{A(r)}(\omega)$ the total time spent by σ_ω when attacked by A, which in turn uses coins r. Likewise, denote by $n_{A(r)}(\omega)$ the number of interactions of A with σ_ω, when A uses coins r. By the hypothesis that A is a q-reset attack, for every value of ω, it holds that $\mathrm{E}_r[n_{A(r)}(\omega)]$ is upper-bounded by $q()$. On the other hand, $t_{A(r)}(\omega) \stackrel{\text{def}}{=} T_{A(r)}(\omega)/n_{A(r)}(\omega)$ corresponds to the (average) time spend by σ_ω in a single iteration with $A(r)$. Thus, if σ satisfies Definition 4, then $\mathrm{E}_\omega[\max_r\{t_{A(r)}(\omega)\}]$ is upper-bounded by some polynomial $p()$, because the attack of $A(r)$ on a single iteration of σ_ω can be emulated in the model of Definition 4. Indeed, the last assertion is the core of the entire argument. Now, observing (see details below) that $\mathrm{E}_{r,\omega}[T_{A(r)}(\omega)] = \mathrm{E}_{r,\omega}[n_{A(r)}(\omega) \cdot t_{A(r)}(\omega)]$ is upper-bounded by the product of $\max_\omega\{\mathrm{E}_r[n_{A(r)}(\omega)]\}$ and $\mathrm{E}_\omega[\max_r\{t_{A(r)}(\omega)\}]$, and using the foregoing upper-bounds, it follows that σ satisfies Definition 3. Details follows.

Let us first prove that $\mathrm{E}_{i,j}[a_{i,j}b_{i,j}]$ is upper-bounded by $\max_j\{\mathrm{E}_i[a_{i,j}]\} \cdot \mathrm{E}_j[\max_i\{b_{i,j}\}]$. This fact can be proved by noting that $\mathrm{E}_j[\mathrm{E}_i[a_{i,j}b_{i,j}]] \le \mathrm{E}_j[\max_i\{b_{i,j}\} \cdot \mathrm{E}_i[a_{i,j}]]$, letting $B_j = \max_i\{b_{i,j}\}$ and $A_j = \mathrm{E}_i[a_{i,j}]$, and using $\mathrm{E}_j[B_jA_j] \le \max_j\{A_j\} \cdot \mathrm{E}_j[B_j]$. We now apply this fact to the analysis of $\mathrm{E}_{r,\omega}[T_{A(r)}(\omega)]$, obtaining

$$\mathrm{E}_{r,\omega}\left[T_{A(r)}(\omega)\right] \;=\; \mathrm{E}_{r,\omega}\left[n_{A(r)}(\omega) \cdot t_{A(r)}(\omega)\right]$$

[20] Note that, unlike in Definitions 1-3, the attacker is given σ's auxiliary input (i.e., z). This is most natural in the context of the current attack, which is also given σ's internal coin tosses (i.e., ω).

$$\leq \max_{\omega}\left\{\mathrm{E}_r\left[n_{A(r)}(\omega)\right]\right\}\cdot \mathrm{E}_{\omega}\left[\max_{r}\left\{t_{A(r)}(\omega)\right\}\right]$$

which is in turn upper-bounded by $q()\cdot p()$.

To show that the foregoing containments are strict we present corresponding strategies that witness the separations. The following examples are rather minimal, but they can be augmented into strategies that seem natural (even for natural protocols). For example, a strategy that halts immediately upon receiving the message 0 and runs for exponential time upon receiving the message 1 witnesses the separation between Definition 1 and Definition 2, when assuming that the designated machine M_0 (of Definition 1) always sends the message 0. Note that this example has nothing to do with the issue of expected polynomial-time (although an example that does relate to the latter issue can be constructed similarly).

To separate Definition 3 from Definition 4 consider a strategy that uniformly selects an n-bit long string r, and upon receiving a message s halts immediately if $s \neq r$ and halts after making 2^n steps otherwise. Clearly, this strategy does not satisfy Definition 4, but it does satisfy Definition 3.

A small twist on the foregoing example can be used to separate Definition 2 from Definition 3: Suppose that upon receiving s, the strategy first sends r, and then halts immediately if $s \neq r$ and halts after making 2^n steps otherwise. In this case a 2-reset attack can cause this strategy to always run for 2^n steps, while no ordinary interactive machine can do so. ■

Discussion: Definitions 2–4 versus Definition 1. We believe that there is a fundamental difference between Definitions 2–4 on the one hand and Definition 1 on the other hand. This can be demonstrated by considering *strict* PPT versions of all four definitions; that is, versions of these definition in which each bound on an expectation is replaced by a corresponding strict bound (i.e., a bound that holds with probability 1). Then, the resulting (strict) versions of Definitions 2–4 coincide,[21] but remain separated from the (strict) version of Definition 1 (as actually shown in the proof above). Note that the (strict) version of Definition 1 is extremely sensitive to minuscule variations in the probabilistic behavior of the designated machine M_0 (i.e., variations that change the support). We believe that the combination of these facts speaks against Definition 1.

3 Results for Zero-Knowledge

The setting of zero-knowledge provides a good warm-up for the general study of secure protocols. Recall that, in the context of zero-knowledge, simulators are used to establish the security of predetermined prover strategies with respect to attacks by adversarial verifiers. We start by showing that (normal black-box) simulators that handle strict PPT adversaries also handle adversaries that are expected PPT (under Definitions 3 and 4). We next turn to an expected PPT version of the standard sequential composition theorem. (In Section 4, analogous results are proved for general secure protocols.)

Since the notion of *normal black-box simulators* is pivotal to our results, let us start by briefly recalling the standard definition of *black-box simulators* (see, e.g., [G01, Def. 4.5.10]). Loosely speaking, a black-box simulator is a universal machine that is given oracle access to a deterministic strategy and provides a simulation of the interaction of this strategy with the party attacked by this

[21] Consider a (standard) interactive machine that guesses at random the internal coin tosses of σ. Then, the strict version of Definition 2 guarantees that even in the rare case that this guess is correct, the strategy σ makes only a polynomial number of steps. But this conditional probability space is exactly the probability space that occurs in Definition 4, which implies that σ satisfies the (strict) version of Definition 4.

strategy.[22] In extending this notion to randomized strategies, we refer to providing the simulator with oracle access to a residual (deterministic) strategy obtained by fixing random coin tosses to the given randomized strategy.

Typically, one considers the execution of black-box simulator when given oracle access to any (strict or expected) PPT adversary. In that case, one sometimes states both the complexity and the quality of the simulation when referring only to the case that the oracle is a PPT strategy.[23] While the restriction of the quality requirement to the said case is often essential, this is typically not the case with respect to the complexity requirement (which refers only to the number of steps taken by the black-box machine itself). Indeed, it is more natural to formulate the complexity requirement when referring to any possible oracle. We adopt this convention below, but in order to avoid possible confusion (with different views) we refer to simulators that satisfy this convention as normal.

Definition 6 (normal black-box simulators): *A black-box simulator is called* normal *if, on any input and when given oracle access to* any strategy*, it makes an* expected *number of steps that is upper-bounded by a fixed polynomial in the length of the input,* where each oracle call is counted as a single step*. That is, there exists a polynomial q such that the* expected *number of steps made by the simulator itself, on any input x and oracle access to any strategy σ, is at most $q(|x|)$.*

Although it is possible to construct black-box simulators that are not normal (e.g., they run for exponential time if the black-box manages to solve a hard problem), the standard black-box simulators (e.g., the ones of [GMR, GMW, GK96]) are all normal. Furthermore, normality seems a very natural property and *it is easy to verify.* For example, if the running-time analysis of a simulator (unlike the analysis of the quality of its output) does not rely on any intractability assumptions, then it is probably the case that the simulator is normal.[24]

The total simulation time. We will often refer to the (total) simulation time of the combined simulator S^{V^*}, which consists of a (normal) black-box simulator S that is given oracle access to an

[22]In typical use of a black-box simulator one also refers to the quality of this simulation. Specifically, it is require that if the former strategy is efficient (in some adequate sense), then the simulation is computationally indistinguishable from the real corresponding interaction. Since the notion of efficiency will vary (i.e., from strict PPT to expected PPT), we shall separate the operational aspect of the black-box simulator from the quality of the output that it produces; that is, we shall discuss each aspect separately (rather than coupling them together).

[23]Even in the case that the complexity requirement is confined to the case that the simulator accesses an arbitrary (strict or expected) PPT adversary, one may distinguish between two requirements regarding the complexity of the simulation. The more liberal requirement, which is rarely used, only mandates that for any such adversary the total simulation time (see below) must be feasible. This means that the total simulation time may be bounded by an arbitrary polynomial that is not necessarily linearly related to the (polynomial) running-time of the adversary. A more restricted and natural formulation, which is typically used, either refers (only) to the number of steps taken by the simulator itself or views oracle calls as single steps (i.e., counting them at unit cost). Specifically, the number of steps of the black-box simulator itself is bounded by a fixed polynomial, regardless of the (polynomial) complexity of the strategy to which it is given oracle access. (Indeed, in such a case, the total simulation time is linearly related to the running-time of the adversary.) Definition 6 takes this approach to its logical conclusion by requiring that, given oracle access to any strategy (regardless of its complexity), the (expected) number of steps taken by the black-box simulator itself is bounded by a fixed polynomial. Indeed, Definition 6 reflects the intuition that the operational aspects of a black-box simulator (unlike the quality of its output) should not be affected by the specifics of its oracle. Note that the gap between the foregoing restricted formulation (which only refers to PPT strategies) and Definition 6 (which refers to all strategies) can be easily bridged if the number of steps taken by the black-box simulator itself is *strictly* polynomial (rather than having *expectation* that is bounded by a polynomial).

[24]The word "probably" indicates that the said implication is not claimed as a fact but is rather suggested as a rule of thumb and/or as a conjecture regarding any natural case.

adversarial verifier V^*. Needless to say, for any normal simulator S, if V^* is *strict* PPT then the *expected* (total) simulation time of S^{V^*} is polynomial. As observed by Katz and Lindell [KL08], this is not necessarily the case if V^* is *expected PPT w.r.t Definition 2.* The key observation, which motivates Definition 3, is that the desired bound on the *expected* (total) simulation time of S^{V^*} does hold if V^* is *expected PPT w.r.t any reset attack.*

Observation 7 *If S is a normal black-box simulator and V^* is* expected polynomial-time w.r.t Definition 3*, then the expected total simulation time of S^{V^*} is polynomial.*

Proof: Since S is a normal black-box simulator, there exists a polynomial q such that, for every setting of coins ω for V^*, it holds that the expected number of times that S invokes the residual strategy V^*_ω is upper-bound by $q()$. Thus, S is a q-reset attack on V^*. Since V^* satisfies Definition 3, it follows that the expected (total) number of steps taken by V^* during the entire simulation is upper-bound by a polynomial. The claim follows. ■

The (knowledge) tightness lens. The foregoing proof clarifies that the (polynomial) upper-bound q guaranteed for a normal simulator (in Definition 6) provides a bound on the knowledge tightness established by this simulator. Indeed, the polynomial p in Definition 3 should be viewed as an upper bound on the expected running-time of the relevant strategy (as per Definition 3).[25] Thus, the foregoing proof shows that the interaction of any expected p-time (w.r.t Definition 3) verifier V^* is simulated in total expected time $q() \cdot p()$, which implies that this normal simulator has tightness $1/q$. We mention that better tightness bounds may be obtained when using a more refined definition of normality in which both the (expected) number of simulator steps and the (expected) number of oracle calls are bounded. Denoting the first polynomial bound by q_0 and the second polynomial bound by q, the proof of Observation 7 yields a bound of $q() \cdot p() + q_0()$ on the total (expected) simulation time, and a knowledge tightness bound of $1/q$ follows.

3.1 Simulating expected PPT adversaries

Bearing in mind that (in the context of zero-knowledge) the simulator is a standard algorithm, it suffices to state the following result with respect to Definition 3, and its applicability to Definition 4 follows as a special case.

Theorem 8 (extendability of normal black-box simulators, the zero-knowledge case): *Let (P, V) be an interactive proof* (or argument) *system for a set L, and $\langle P, V^*\rangle(x)$ denote the output of the adversarial verifier strategy V^* on input x after interacting with the prescribed prover P. Let M be a normal black-box simulator that, on input in L and when given access to any* strict *PPT strategy V^*, produces output that is computational indistinguishable from $\langle P, V^*\rangle$. Then, when M is given oracle access to any strategy V^* that is* expected PPT w.r.t any reset attack*, the expected simulation time of M^{V^*} is polynomial and the output is computational indistinguishable from $\langle P, V^*\rangle$.*

Note that the hypothesis allows the simulator to run in expected PPT while simulating a strict PPT adversary. This makes the hypothesis weaker and the theorem stronger; that is, the theorem can be applied to a wider class of protocols (including protocols that are not known to have strict PPT simulators such as, e.g., the constant-round zero-knowledge proof of [GK96]).

[25] Indeed, it is telling to note that if a strategy is expected p-time w.r.t Definition 4, then p can be used as a bound in Definition 3 (see the proof of Proposition 5).

Proof: Fixing any expected PPT w.r.t Definition 3 strategy V^*, we first note that (by Observation 7) the expected simulation time of M^{V^*} is polynomial. To analyze the quality of this simulation, suppose towards the contradiction that D distinguishes between the simulation and the real interaction, and let p be a polynomial such that the distinguishing gap of D for infinitely many $x \in L$ is at least $\epsilon(|x|) \stackrel{\text{def}}{=} 1/p(|x|)$. Let $t^*(x)$ denote the total (over all invocations) expected number of steps taken by V^* when invoked by M. Note that $t^*(x)$ is upper-bounded by a polynomial in $|x|$, and assume (without loss of generality) that $t^*(x)$ also upper-bounds the expected running time of V^* in the real interaction (with P). Now, consider a *strict* PPT V^{**} that emulates V^*, while truncating the emulation as soon as $3t^*/\epsilon$ steps are emulated. Then, the variation distance (a.k.a statistical difference) between $M^{V^*}(x)$ and $M^{V^{**}}(x)$ is at most $\epsilon(|x|)/3$, because $\epsilon/3$ upper-bounds the probability that the total number of steps taken by V^* during all invocations by M exceeds $3t^*/\epsilon$ (and otherwise V^{**} perfectly emulates all these invocations, since none exceeds $3t^*/\epsilon$ steps). Similarly, the variation distance between $\langle P, V^* \rangle(x)$ and $\langle P, V^{**} \rangle(x)$ is upper-bounded by $\epsilon(|x|)/3$. It follows that D distinguishes the simulation $M^{V^{**}}$ from the real interaction $\langle P, V^{**} \rangle$ with a gap that exceeds $\epsilon/3$, on infinitely many inputs in L, in contradiction to the hypothesis that M simulates all *strict* PPT verifiers. ■

Digest. We believe that the fact that the proof of Theorem 8 is rather straightforward should not be counted against Definition 3, but rather the other way around. That is, we believe that the claim that the simulation of strict PPT adversaries extends (without modifications) to expected PPT adversaries is natural, and as such a good definition of expected PPT adversaries should support it. It may be that Theorem 8 can be generalized also to arbitrary black-box simulators and even to arbitrary universal simulators, but the current proof fails to show this: the running-time analysis relies on the hypothesis that the simulator is normal, whereas the output-quality analysis relies on the hypothesis that the simulator is black-box.[26] While this possibility is certainly interesting, we consider it secondary to the main message carried by Theorem 8 (i.e., that a good definition of expected PPT strategies (such as Definition 3) supports the "extendability of simulators" from handling strict PPT verifiers to handling expected PPT verifiers).[27]

Note that the combined simulator resulting from Theorem 8 is trivially expected PPT under reset attacks (and also under Definition 4), because it is a non-interactive machine (which runs in expected polynomial-time). Things are not as simple when we move to the setting of secure protocols, where the simulator is an interactive strategy (which operates in a so-called ideal-model). See Section 4.1.

The (knowledge) tightness lens. Note that the knowledge tightness (as provided by Theorem 8) does not change when moving from strict PPT verifiers to expected PPT verifiers.

[26] Recall that a universal simulator obtains the code of the adversary's strategy rather than a black-box access to it. Thus, it may be the case that such a simulator can distinguish the code of V^* from the code of V^{**} (i.e., the timed version of V^*), and produce bad output in the latter case. Indeed, a "natural" simulator will not do so, but we cannot rely on this. Turning to a more natural example, we note that the known non-black-box simulator of Barak [B01] (as well as its modification [BG08]) may fail to simulate expected PPT verifiers, because the random variable representing its simulation time is polynomially related (rather than linearly related) to the running-time of the verifier. Recall that it may be the case that $t(x)$ has expectation that is upper-bounded by a polynomial in $|x|$ while $t(x)^2$ has expectation that is lower-bounded by $\exp(|x|)$; for example, consider $t : \{0,1\}^* \to \mathsf{N}$ such that $\Pr[t(x) = 2^{|x|}] = 2^{-|x|}$ and $\Pr[t(x) = |x|^2] = 1 - 2^{-|x|}$.

[27] One of the reviewers asked us to state an opposite opinion by which Theorem 8 is mainly due to the normality hypothesis. We strongly disagree with this opinion. In particular, we mention that the simulator in [GK96] is normal, but (as shown in [KL08]) it fails to handle some verifier strategies that are expected PPT w.r.t Definition 2.

3.2 Sequential composition

The following theorem (i.e., Theorem 9) is an expected PPT version of the standard result (of [GO94]) that refers to *strict* PPT adversaries and simulators (see also [G01, Lem. 4.3.11]). Note that the standard result does not require the simulator to be black-box (let alone normal). The reason for the extra requirement in Theorem 9 will become clear in the proof.

Theorem 9 (expected PPT version of sequential composition for zero-knowledge:) *In this theorem zero-knowledge means the existence of a normal black-box simulator that handles any* expected *PPT w.r.t Definition 3* (resp., w.r.t Definition 4) *adversarial verifier, where handling means that the corresponding combined simulator runs in* expected *PPT and produces output that is computationally indistinguishable from the real interaction. Suppose that (P, V) is a zero-knowledge protocol. Then, sequentially invoking (P, V) for a polynomial number of times yields a protocol, denoted (P', V'), that is zero-knowledge.*

We stress that Theorem 9 differs from the standard result (of [GO94]) in two ways. Theorem 9 refers to expected PPT adversarial verifiers (rather than to strict PPT ones), and it refers to expected PPT simulators (rather than to strict PPT ones).

Proof: The proof of the strict PPT version (see [G01, Sec. 4.3.4]) proceeds in two steps: First, any verifier V^* that attacks the composed protocol (or rather the prover P') is transformed into a verifier V^{**} that attacks the basic protocol (or actually the prover P). This transformation is quite straightforward; that is, V^{**} handles a single interaction with P (while receiving the transcript of previous interactions as auxiliary input). Let M denote a simulator for (P, V^{**}). Then (in the second step), a simulator for the composed protocol (or rather for the attack of V^* on P') is obtained by invoking M for an adequate number of times (using a correspondingly adequate auxiliary input in each invocation).

Wishing to pursue the foregoing route, we merely need to check that any verifier V^* that is expected PPT w.r.t Definition 3 (resp., Definition 4) is transformed into a verifier V^{**} that is expected PPT w.r.t Definition 3 (resp., Definition 4). Unfortunately, this is not necessarily the case. Indeed, the expected running-time of V^{**} when given a *random* auxiliary input (i.e., one produced at random by prior interactions) is polynomial, but this does not mean that the expected running-time of V^{**} *on each possible value* of the auxiliary input is polynomial. For example, it may be the case that, with probability $2^{-|x|}$ over the history of prior interactions, the current interaction of V^* (i.e., V^{**} with the corresponding auxiliary input) runs for $2^{|x|}$ steps. The bottom-line is that V^{**} may not be expected PPT w.r.t any reasonable definition (let alone w.r.t Definition 3 or Definition 4).

In view of the foregoing, we take an alternative route. We only use the hypothesis that some normal black-box simulator M can handle all *strict* PPT verifiers that attack the basic prover P. Still, the hypothesis provides us with expected PPT simulators (rather than with strict PPT ones). Nevertheless, we observe that the proof of [G01, Lem. 4.3.11] (i.e., the strict PPT version) can be extended to the case that the simulation of the basic protocol (w.r.t *strict* PPT adversaries) runs in *expected* PPT. The key observation is that in this case V^{**} is strict PPT, although it will be fed with auxiliary inputs that are produced in expected PPT (by the simulation of prior interactions of V^{**} with P). Thus, following the construction in the proof of [G01, Lem. 4.3.11], we obtain an *expected* PPT simulation that handles any *strict* PPT attack on P'. Furthermore, the simulation amounts to invoking M for a polynomial number of times (while providing it with black-box access to V^{**}, which in turn is implemented by a black-box access to V^*). It follows that the simulation of (P', V^*) is performed by a normal black-box simulator (because M is normal). Hence, we have obtained a *normal black-box simulator that can handle any strict PPT attack on the composed protocol* (or rather on the prover P'). The current theorem follows by applying Theorem 8 to the latter simulator. ■

Digest. The proof of Theorem 9 is somewhat disappointing because it does not use the hypothesis that P is zero-knowledge w.r.t *expected* PPT verifiers. Instead, Theorem 8 is used to bridge the gap between strict and expected PPT verifiers. A similar (but not identical) phenomenon will occur in the sequential composition theorem for general protocols, presented in Section 4.2.

The (knowledge) tightness lens. The normal simulator constructed for (P', V') in the proof of Theorem 9 preserves the knowledge tightness of the original simulator (i.e., the normal simulator provided for (P, V)).

4 Results for General Secure Protocols

In this section we extend the treatment of zero-knowledge (provided in Section 3) to a treatment of arbitrary secure protocols. The extension is quite straightforward, once the key notions are properly extended. The main issue that deserves attention is that, in the context of arbitrary secure protocols, simulators are not standard algorithms but rather interactive strategies (for a corresponding ideal-model – to be discussed next). Consequently, notions such as expected PPT simulation and normal (black-box) simulators will have to be clarified. For simplicity, we focus on the two-party case.

Recall that the standard ("simulation-based") definitions of secure protocols call for comparing the real execution of the protocol (when certain parties are controlled by an adversary) to the affect of a corresponding adversary in an *ideal model* (see, e.g., [G04, Sec. 7.2]). The ideal model consists of the parties sending their inputs to a trusted party that provides each party with its corresponding output, where the trusted party computes these outputs according to the predetermined functionality that the protocol is supposed to securely compute. Thus, the actions of the adversary in the ideal model are confined to selecting the messages sent to the trusted party (by the parties controlled by the adversary) and computing its final output based on the messages it received from the trusted party (i.e., the messages received by the parties controlled by the adversary). In the two-party case, this adversary sends a single message to the trusted party and receives a single message in return. Note that this adversary is an interactive machine, although its interaction is very minimal, and thus the various definitions of *expected PPT strategies* should and can be applied to it.

Another point to note is that the ideal-model adversary is viewed as a simulator of the real-model adversary, and that (as in the case of zero-knowledge) the simulator is typically described as a universal machine that is given black-box access to the real-model adversary that it simulates. For simplicity, we shall refer to the ideal-model adversary as *the simulator* and to the real-model adversary as *the adversary.*

Turning to the notion of *normal black-box simulators*, let us first restate Definition 6 (which refers to non-interactive simulators). For any black-box simulator S and any adversary A, we consider an *imaginary machine I that emulates S^A such that each oracle call to A is emulated in unit time.* Then, Definition 6 mandates that, for every adversary A, the corresponding I is expected PPT.[28] In our context, the simulator itself is an interactive machine, and thus the imaginary

[28] Thus, we have restated the condition that refers to the number of steps performed by S itself (when using oracle

machines will also be interactive. For $i = 1,2,3,4$, we say that a black-box simulator S is normal w.r.t Definition i if, for every adversary A, the corresponding $I = S^A$ is expected PPT w.r.t Definition i. We note that natural simulators used in security proofs are normal. This holds for simulators of simple protocols (cf., e.g., [G04, Sec. 7.4.3.1-7.4.3.3]) as well as for simulators of complex protocols obtained by composition (cf., e.g., [G04, Sec. 7.4.4]).

Clarification. As usual, we shall only consider the secure computation of functionalities that can be (insecurely) computed in strict PPT. Similarly, we shall only consider protocols in which all *prescribed strategies* are implementable in strict PPT; that is, the proper execution of all protocols (by honest users) only requires strict PPT. Indeed, the notion of expected PPT is only applied to adversaries (and to simulators, which are themselves ideal-model adversaries). Needless to say, this attitude is in perfect agreement with the views expressed in the introduction.

4.1 Simulating expected PPT adversaries

In continuation to Section 3.1, we prove that normal black-box simulation of strict PPT adversaries can be extended to expected PPT adversaries. Unlike in Theorem 8, here the result (i.e., Theorem 10) is stated for both the new definitions, because the combined simulator is an interactive machine (and thus Definitions 3 and 4 do not necessarily coincide when applied to it).

Theorem 10 (extendability of normal black-box simulators, the case of general two-party protocols): *Let* Π *be a two-party protocol and* $\text{REAL}_A(\overline{x})$ *denote the output of its execution, on input tuple* $\overline{x}$*, under an attack of the adversary* A*. Let* S *be a normal w.r.t Definition 3* (resp., Definition 4) *simulator and* $\text{IDEAL}_F^A(\overline{x})$ *denote the output of its execution, on input tuple* $\overline{x}$*, oracle access to the strategy* A*, and when the trusted party answers according to the functionality* F*. Suppose that for every* strict *PPT strategy* A*, it holds that* IDEAL_F^A *is computational indistinguishable from* REAL_A*. Then, for every strategy* A *that is* expected PPT w.r.t Definition 3 (resp., Definition 4), *the total simulation time of the combined simulator* S^A *is* expected PPT w.r.t Definition 3 (resp., Definition 4) *and* IDEAL_F^A *is computational indistinguishable from* REAL_A.

As in case of zero-knowledge, Theorem 10 asserts that known simulators that handle strict PPT adversaries can also handle adversaries that run in expected polynomial-time under the new definition(s). (Again, this holds even if the former simulators run in expected PPT.)

Proof: The current proof is analogous to the proof of Theorem 8, except that the verification of the expected total running-time of the combined simulation is slightly less evident. The key point is that a definitional attack (i.e., as in Definitions 3 and 4) on the combined simulator S^A yields a corresponding attack on A, whereas A satisfies Definition 3 (resp. Definition 4) by the hypothesis. Details follow.

We can focus on the total time spent by A in all its invocations by S, since the number of steps of S itself is upper-bounded by the normality hypothesis. Let us first consider the version that refers to Definition 4, denoting by $n_{\omega_A}(\omega_S)$ the maximum number of invocations of A by S, when A (resp., S) uses coins ω_A (resp., ω_S) and the maximization is over all possible messages (supposedly by the trusted party) that can be provided to the simulator (maximized for these choices of ω_A and ω_S). By the normality hypothesis (applied to the residual adversaries A_{ω_A}), it follows that $\max_{\omega_A}\{\text{E}_{\omega_S}[n_{\omega_A}(\omega_S)]\}$ is upper-bounded by a polynomial, denoted q. Turning to a setting in which A interacts with a "magic" machine as in Definition 4, we denote by $t(\omega_A)$ the maximum running

calls to A) as a condition that refers to the total number of steps performed by the imaginary machine I.

time of A (in such an interaction) when the maximization is over all possible messages sent to A (again maximized for this choice of ω_A). It follows that $\mathrm{E}_{\omega_A}[t(\omega_A)]$ is upper-bounded by a polynomial, denoted p (since A is PPT w.r.t Definition 4). Finally, we consider the total time spent by A when S^A interacts with a magical machine (as in Definition 4), and upper-bound it by

$$\begin{aligned}\mathrm{E}_{\omega_S,\omega_A}[n_{\omega_A}(\omega_S)\cdot t(\omega_A)] &= \mathrm{E}_{\omega_A}[\mathrm{E}_{\omega_S}[n_{\omega_A}(\omega_S)]\cdot t(\omega_A)] \\ &\leq \mathrm{E}_{\omega_A}[\max_{\omega}\{\mathrm{E}_{\omega_S}[n_{\omega}(\omega_S)]\}\cdot t(\omega_A)] \\ &= \max_{\omega}\{\mathrm{E}_{\omega_S}[n_{\omega}(\omega_S)]\}\cdot \mathrm{E}_{\omega_A}[t(\omega_A)]\end{aligned}$$

which equals $q()\cdot p()$. This establishes the claim for Definition 4.

Turning to the version that refers to Definition 3, we apply an analogous analysis. Specifically, *fixing any reset attack on the simulator* S^A, we let $n_r(\omega_S,\omega_A)$ denotes the number of invocations of $A(\omega_A)$ by $S(\omega_S)$ when $S^{A(\omega_A)}(\omega_S)$ is invoked by the reset attack that uses coins r. The admissibility of this reset attack on S^A means that, for any ω_A and ω_S, the expected number of invocations of $S^{A(\omega_A)}(\omega_S)$ by this attack is upper-bounded by a polynomial (where the expectation is taken over all possible choices of r). Fixing any ω_A, we may view the foregoing reset attack (on S^A) as a reset attack on $S^{A(\omega_A)}$, and note that it is an admissible reset attack (since, for every ω_S, the expected number of invocations of $S^{A(\omega_A)}(\omega_S)$ is upper-bounded by the aforementioned polynomial). Hence, the normality condition of S (w.r.t Definition 3) implies that the expected number of times that S invokes $A(\omega_A)$ during this attack is upper-bounded by a polynomial, denoted q; that is, $\max_{\omega_A}\{\mathrm{E}_{r,\omega_S}[n_r(\omega_S,\omega_A)]\}$ is upper-bounded by q. Now, combining the reset attack on S^A with S itself, we obtain an admissible reset attack on A (i.e., a q-reset attack on A). Thus, by Definition 3 (applied to A), it follows that the expected total amount of time spent by A in these interactions is upper-bounded by a polynomial. ■

4.2 Sequential composition

In continuation to Section 3.2, we turn to discuss the preservation of the security of general protocols under sequential composition. The formulation is more complex in the current setting, because sequential composition of general protocols refers to a model of oracle-aided protocols (a.k.a "hybrid" model). Thus, we need to extend our definitional treatment of expected PPT to that model.

Recall that an oracle-aided protocol Π that uses oracle calls to a functionality f, is a protocol augmented by special instructions by which the (two) parties may invoke the functionality f (several times). Each invocation is performed by sending inputs to f, via special (imaginary) channels, and receiving corresponding outputs (again via special channels).[29] Thus, in the various definitions of expected PPT we need to refer also to the distribution of the messages obtained through the aforementioned special channels. Specifically, when considering a strategy in the oracle-aided model, the (definitional) attack[30] on this strategy controls both the ordinary channels (on which the strategy expects to get messages from other parties) and the special channels (on which the strategy expects to get outputs from the functionality). We stress that only under (the natural extension of) Definition 1, it is the case that the messages delivered over the special channels must fit the designated functionality f.

[29] We stress that each invocation of f is performed instantaneously and no other protocol activity (i.e., neither an ordinary communication nor another invocation of f) is performed concurrently. As usual, towards the time complexity, each invocation is considered a single step.

[30] Note that here we refer to the attacks used (as a mental experiment) in the various definitions of expected PPT strategies (especially in Definitions 3 and 4).

A sequential composition theorem refers to an oracle-aided protocol that uses oracle calls to some functionality, and to the effect of replacing these oracle calls by invocations of a secure protocol for the said functionality. In the standard results of this type (cf. [C00]), it is assumed that the proof of security of the sub-protocol (which replaces the oracle calls to the functionality) is via a strict PPT simulator. The difficulty addressed here is that allowing an expected PPT simulator for this sub-protocol requires considering expected PPT adversaries for the oracle-aided protocol (even if we only care about strict PPT adversaries for the composed protocol). But if the oracle-aided protocol is secure also with respect to expected PPT adversaries then we are fine (as far as strict PPT adversaries for the composed protocol are concerned). As in the proof of Theorem 9, if all the simulators guaranteed by the hypothesis are normal, then we can extend the result to expected PPT adversaries.

Theorem 11 (expected PPT version of the standard sequential composition theorem[31]:) *In this theorem security means the existence of* normal black-box *simulators that can handle*[32] *any* expected *PPT adversary,* where normality and expected PPT are defined as in either Definition 3 or Definition 4. *Suppose that F can be securely computed by an oracle-aided protocol Π that is given oracle access to the functionality f, which can be securely computed by a standard protocol ρ. Then, F can be securely computed by a standard protocol Π', which is composed of Π and ρ.*

Note that, by Theorem 10, *it suffices to have in the hypothesis expected PPT* (or rather normal black-box) *simulators that can simulate any strict PPT adversary.* Actually, the following proof invokes Theorem 10 anyhow, which in turn is the reason that the definition of security refers to simulators that operate in a black-box and normal fashion.

Proof: As in the proof of Theorem 9, the first idea that comes to mind is adapting the standard proof of the corresponding result (i.e., [G04, Thm. 7.4.3]) that refers to strict PPT. Specifically, the standard proof (as presented, say, in [G04, Sec. 7.4.2]) proceeds as follows: First, any adversary that attacks the standard protocol Π' is transformed into an adversary that attacks the standard protocol ρ. Next, the former adversary (i.e., of Π') as well as a simulator for the latter adversary (i.e., of ρ) are combined and transformed into an adversary that attacks the oracle-aided protocol Π (which uses oracle calls to f). A simulator of this adversary of Π yields the desired simulation.

However, as in the proof of Theorem 9, it is not necessarily the case that if the adversary attacking Π' is expected PPT then the adversary obtained for ρ is also expected PPT. Thus, again, we take an alternative route, starting by establishing the current theorem for strict PPT adversaries attacking Π' and next applying Theorem 10 to extend the result to adversaries that are expected PPT w.r.t Definition 3 (resp., Definition 4). Now there is no problem with the first transformation (which transforms any strict PPT adversary attacking Π' into a strict PPT adversary attacking ρ). Hence, we obtain a simulator for ρ, which runs in expected PPT w.r.t Definition 3 (resp., Definition 4). Combining this simulator with the former adversary (for Π'), we obtain an adversary attacking Π that runs in *expected* PPT according to Definition 3 (resp., Definition 4).

The key point is that (by the hypothesis) we do have a (normal black-box) simulator that can handle any *expected* PPT adversary attacking Π. Thus, proceeding as in the proof of [G04,

[31] This is an expected PPT version of the Sequential Composition Theorem of [C00] (see also [G04, Thm. 7.4.3]), which refers to security as the existence of strict PPT simulators that handle any strict PPT adversary. As in Theorem 9, our expected PPT version requires that the simulators in the hypothesis operate in a black-box (and normal) manner.

[32] As in Theorem 9, handling means that the corresponding combined simulator runs in expected PPT under the relevant definition and produces output that is computationally indistinguishable from the real interaction.

Thm. 7.4.3], we obtain a simulator for Π', which is expected PPT w.r.t Definition 3 (resp., Definition 4). Using the fact that both simulators we used are normal black-box simulators (and so is the construction presented in the proof of [G04, Thm. 7.4.3]), we infer that the simulator obtained for Π' is a normal black-box simulator. This allows invoking Theorem 10, and thus extending the simulation to adversaries that are expected PPT w.r.t Definition 3 (resp., Definition 4). The theorem follows. ■

Digest. Note that the partial result by which Π' is secure w.r.t *strict* PPT adversaries (via an expected PPT simulator) was established using the following two hypotheses: (1) the simulator for Π can handle *expected* PPT adversaries, and (2) the (expected PPT) simulator for ρ can handle *strict* PPT adversaries. That is, this partial result neither uses the hypothesis that the simulator for ρ can handle expected PPT adversaries nor the hypothesis that both simulators operate in a black-box (and normal) fashion. The latter hypothesis is used in order to guarantee that the simulator constructed for Π' is a normal black-box simulator, which in turn is used for extending the partial result to the general result stated in Theorem 11. The hypothesis that the simulator for ρ can handle expected PPT adversaries is never used.

Recall that, as a direct corollary to Theorems 10 and 11, we may obtain the following result, which suffices in many (if not all)[33] applications. This result refers to the composition of protocols that are proved secure with respect to *strict* PPT adversaries by using *expected* PPT simulators.

Corollary 12 (sequential composition for the mixed strict/expected model:) *Here security means the existence of* normal black-box *simulators that can handle any* strict *PPT adversary,* where normality is defined as in either Definition 3 or Definition 4 (and, in particular, allows expected PPT simulators). *Suppose that F can be securely computed by an oracle-aided protocol* Π *that is given oracle access to the functionality f, which can be securely computed by a standard protocol ρ. Then, F can be securely computed by a standard protocol* Π'*, which is composed of* Π *and* ρ*.*

Actually, the simulator for ρ need not be black-box, because Corollary 12 can be derived as an consequence of the aforementioned partial result, which only requires the simulator of Π to handle expected PPT adversaries. The latter condition is guaranteed by applying Theorem 10 to the normal black-box simulator that can handle any strict PPT adversary for Π.

The (security) tightness lens. The proof of Theorem 11 preserves the security tightness of the strict PPT result (i.e., [G04, Thm. 7.4.3]), which in turn is the multiple of the security tightness of the two underlying protocols (i.e., Π and ρ). The same holds with respect to Corollary 12.

4.3 Concurrent composition

Turning to concurrent composition theorems, we recall the pivotal role of environmental security (a.k.a UC-security [C01]) in that context. Specifically, Canetti [C01] put forward a robust notion of security (i.e., environmental security), and proved that any protocol that satisfies this notion also preserves security under arbitrary concurrent executions. Since environmental security refers to a single execution, an appealing methodology for providing protocols that are secure under arbitrary concurrent executions emerged: design your protocol to be environmentally secure and obtain (for free) security under concurrent executions. Our goal is to extend this methodology, which was developed for the strict PPT setting, to the expected PPT setting. This requires (1) showing that

[33]The stronger statement relies on the opinions expressed in Section 1.5.

environmental security in the strict PPT setting implies environmental security in the expected PPT setting, and (2) verifying that Canetti's proof extends to the expected PPT setting. But let us start by recalling Canetti's notion of environmental security [C01] (see also [G04, Sec. 7.7.2]), while confining ourselves to standard (non-reactive) functionalities.[34]

A brief introduction to environmental security. Loosely speaking, environmental security[35] is aimed at representing the preservation of the protocol's security when executed within any (feasible) *environment*. The notion of an environment is a generalization of the notion of an auxiliary-input; that is, the environment is an auxiliary oracle (or rather a state-dependent oracle) that the adversary may access. In particular, the environment may represent other executions of various protocols that are taking place concurrently (with the execution that we consider). We stress that the environment is not supposed to assist the proper execution of the protocol (and, in fact, honest parties merely obtain their inputs from it and return their outputs to it). In contrast, the environment may assist the adversary in attacking the protocol. Following the simulation paradigm, we say that a protocol (for computing a functionality F) is environmentally-secure if any feasible *real-model adversary attacking the protocol, with the assistance of any feasible environment*, can be simulated by a corresponding *ideal-model adversary that uses the same environment* (and communicates with a trusted party that represents F). We stress that both adversaries interact with an environment that is selected after they are fixed (i.e., they "use" the environment in a black-box manner). For sake of simplicity, the environment is also responsible for providing the parties with inputs and for trying to distinguish the real-model execution from the ideal-model execution. In the standard formulation (see [G04, Sec. 7.7.2]), the environment is implemented by a (non-uniform) family of polynomial-size circuits (or, equivalently, by strict PPT with arbitrary auxiliary inputs). As usual, the real-model and ideal-model adversaries are modeled as strict PPT interactive machines.

The expected PPT version. Firstly, we apply our definitions of expected PPT (i.e., Definitions 3 and 4) to the real-model and ideal-model adversaries, hereafter referred to as *adversaries* and *simulators* respectively. Note that the (definitional) attacks on these strategies control both the ordinary channels (on which such a strategy expects to get messages from other parties) and the channels used for communication with the environment. Secondly, we apply our definitions of expected PPT (i.e., Definitions 3 and 4) to the environment itself, which after all is merely a strategy.[36] Lastly, we extend the notion of normal black-box simulators such that its "net" time bound (i.e., counting only its own steps) refers to interaction with *any environment*.

Theorem 13 (extendability of simulators, the case of environmental security): *In this theorem, an expected PPT strategy is one that satisfies Definition 3* (resp., Definition 4). *Suppose that* Π *is*

[34]Recall that a (non-reactive) functionality is a randomized version of a multi-input multi-output function (cf. [G04, Sec. 7.2.1]). In contrast to our approach, Canetti's exposition of environmental security [C01] is dominated by reactive functionalities, which are of natural (secondary) interest also when the basic notion of (stand-alone) security is concerned (cf. [G04, Sec. 7.7.1.3]). We see no reason to couple the treatment of environmental security with reactive functionalities.

[35]The term used by Canetti [C01] is *Universally Composable*, abbreviated UC-secure, but we believe that a reasonable sense of "universal composability" is merely a corollary of the suggested definition. Furthermore, as indicated by subsequent research (e.g., [L09]), it is beneficial to distinguish the desired "universal composability" property from the specific way it is formulated.

[36]In fact, since the simulator cannot "rewind" the environment, we may allow the environment to be expected PPT according to Definition 2. However, in the main application (i.e., Theorem 14) we shall only use environments that are expected PPT according to Definition 3 (resp. Definition 4).

environmentally secure in the sense that for every strict *PPT adversary there exists an* expected *PPT simulator such that, for every* strict *PPT environment, the corresponding real-model and ideal-model executions are computationally indistinguishable. Further suppose that the simulator runs in* expected *PPT even when interacting with an arbitrary environment. Then, there exists a normal black-box simulator such that, for every* expected *PPT adversary and every* expected *PPT environment, the following holds:*

1. *The expected* total simulation time *is polynomial, where the* total simulation time *includes the steps taken by the simulator itself, the steps taken by the black-box adversary in all invocations, and all steps taken by the environment.*
2. *The corresponding real-model and ideal-model executions are computationally indistinguishable.*

Note that the hypothesis allows the simulator to run in expected PPT while simulating a strict PPT adversary and that the simulation is guaranteed to be computationally indistinguishable with respect to strict PPT environments. Unlike in the previous extendability theorems (i.e., Theorems 8 and 10), here we did not require the simulator to use the adversary in a black-box manner, because without loss of generality (in the environmental setting) it suffices to consider a fixed (and rather trivial) adversary (cf. [C01]). We did require, however, that the simulator of that adversary runs in *expected* PPT when interacting with any environment (which means that it is "normal w.r.t the environment").

Proof: By the last comment, the hypothesis actually yields a *normal black-box* simulator that handles any *strict* PPT adversary and any *strict* PPT environment. Proceeding as in the proof of Theorem 10, which in turn builds on the proof of Theorem 8, we note that the same simulator can handle any *expected* PPT adversary and any *expected* PPT environment. The current theorem follows. ■

Security under concurrent executions. For any protocol Π, we wish to consider numerous executions of Π that take place concurrently, where the scheduling of messages in the various executions is up to the adversary.[37] In addition, other numerous executions of other protocols (sometimes referred to as "arbitrary network activity") can take place concurrently, but our concern is with the security of the copies of Π. Loosely speaking, this should mean that these actual executions of Π can be simulated in a corresponding ideal-model (where a trusted party answers according to the desired functionality). Needless to say, the simulator control the same parties that are controlled by the adversary in the real-model. For simplicity, consider the case that all executions of the (two-party) protocol Π are played by the same pair of parties (and that the adversary controls a single party).

Canetti [C01] proved that if Π is environmentally secure then the concurrent execution of multiple copies of Π is secure, where security refers to strict PPT adversaries and simulators (as well as such environments when relevant). Loosely speaking, Canetti's proof consists of *simultaneously* replacing all the (real-model) concurrent executions by copies of the simulator (of the environmental security hypothesis) while emulating the adversary's attack on the concurrent system by using the channels of the corresponding environments. (A hybrid argument that refers to partial replacements

[37]Note that this differs from sequential composition (treated in Section 4.2) in that these executions take place concurrently rather than sequentially. Furthermore, additional activity (which is referred to next) takes place concurrently rather than before and/or after these executions.

of real executions by simulations is used for showing that the behavior is maintained.) Here we claim an expected PPT version of Canetti's result.

Theorem 14 (environmental security implies concurrent composability, an expected PPT version (roughly stated)): *Suppose that* Π *is environmentally secure with respect to adversaries, simulators and environments that are* expected PPT w.r.t Definition 3 (resp., Definition 4). *Further suppose that the simulator runs in* expected *PPT even when interacting with an arbitrary environment. Then the concurrent execution of polynomially many copies of* Π *is secure with respect to adversaries and simulators that are* expected PPT w.r.t Definition 3 (resp., Definition 4).

The proof is analogous to the proof of Theorem 11. For clarity, we start by defining an imaginary protocol Π' that consists of polynomially many concurrent copies of Π, each initiated by any party at any time and proceeding at arbitrary pace (i.e., at each time, each party decides whether to initiate a new copy or advance an active copy by sending a corresponding message). Next, adapting the proof of Canetti [C01], we first prove a partial result in which we only consider an arbitrary *strict* PPT adversary that attacks Π' (i.e., polynomially many copies of Π). We note that the simulator constructed by Canetti (for Π') uses the simulator for environmental security of Π in a black-box and normal manner. Thus, the former simulator runs in expected PPT provided that the latter simulator runs in expected PPT, which is definitely the case when simulating residual adversaries and environments that are derived from the *strict* PPT adversary that attacks Π'. Finally, proceeding as in the proof of Theorem 11, we extend the result to any *expected* PPT adversary that attacks Π'. Theorem 14 follows.

5 Alternatives to expected PPT

In standard algorithmic settings, strict PPT captures the intuitive notion of efficient probabilistic computations. However, as explained in Section 1.5, in some cases strict PPT is slightly too rigid and one may seek a more flexible alternative. Expected PPT provides such a flexible alternative, and in fact it is the first such alternative that comes to mind. Throughout this work, we ignored the question of what is a good flexible definition of "efficient probabilistic" algorithms. We merely assumed that it is provided by expected PPT, and focused on extending this notion to interactive machines. In this section we discuss several alternatives to the association of efficient probabilistic algorithms with expected PPT.

Recall that expected PPT *refers to the expected running-time and requires that this expectation be upper-bounded by a polynomial* (in the length of the input). However, as advocated by Levin [L86] in a somewhat different context (see [G97]), a better definition of "flexible probabilistic efficiency" is obtained by *requiring that the running-time itself, as a random variable, be upper-bounded by a polynomial in a random variable that has expectation that is at most linear* (in the length of the input). In particular, Levin's definitional approach eliminates the technical difficulties exemplified at the end of Footnote 26, and provides a robust definition of probabilistic efficiency; that is, if a probabilistic algorithm is deemed "efficient" then also a modification that squares its running time will yield an "efficient" algorithm.[38] In Section 5.1 we extend Levin's definitional approach to interactive strategies, pursuing the same alternatives as those presented in Section 2, and establishing analogous extendability and composition results.

[38]Indeed, this guarantees that Barak's non-black-box simulator [B01] (when applied to "efficient" verifiers) remains "efficient" and an extension of Barak's result to "efficient" strategies follows as in the proof of Theorem 8 (while noting that this specific simulator is not affected by replacing of the code of V^* with the code of V^{**}). For details, see Section 5.1.

In general, the question of how to define flexible probabilistic efficiency for non-interactive algorithms is quite orthogonal to the issues discussed in the current paper (i.e., how to extend such a definition to interactive strategies). Indeed, it seems that any reasonable definition for algorithms can be extended in analogous ways to interactive strategies. For example, in the context of zero-knowledge, it was suggested (cf. [DNS]) to use simulators that, for every desired noticeable deviation ϵ (from the real interaction), run in time that is strictly bounded by a polynomial in $1/\epsilon$. An alternative suggestion (of Vadhan [V06]) is allowing (standard) simulation with varying running-time such that the probability that the simulation takes more than t steps is upper-bounded by $\text{poly}() \cdot t^{-\Omega(1)} + \mu()$, where μ is a negligible function. Note that, in both cases, the definition (stated here for standard algorithms) will have to be extended to interactive machines, and the issues and approaches presented in this paper will apply. For details see Section 5.2.

Before discussing these alternatives in greater detail, we note that all these alternative definitions of "flexible probabilistic efficiency" are (intentionally) more permissive than the standard definition (i.e., expected PPT). We believe that *in the current context*, where expected PPT is reluctantly introduced to account for "probabilistic efficiency" that goes beyond strict PPT, the approach of *restricting probabilistic efficiency to expected PPT is more adequate.*

5.1 Extending Levin's approach

Let us first spell out Levin's suggestion (which was loosely stated above). This suggestion is rooted in the realization that an important aspect of (deterministic and strict probabilistic) polynomial-time as a model of efficient computation is the closure of polynomial-time under natural algorithmic compositions. This feature, in turn, boils down to closure properties of the set of polynomials (i.e., their closure to addition, multiplication, and composition). The problem is that, in the case of "flexible efficient probabilistic computation", directly *upper-bounding the expected running-time* (as underlying the definition of expected PPT) does not provide closure under natural algorithmic compositions. But *upper-bounding the expectation of a root of the running-time* does deliver the desired property. Specifically, we obtain the following definition.

Definition 15 (flexible probabilistic efficiency, following Levin [L86]): *For a probabilistic algorithm A and any string x, let $T_A(x)$ denote a random variable representing the running-time of A on input x. Such an algorithm is said to be* efficient *if there exists a constant $\gamma > 0$ such that for every x it holds that $\mathrm{E}[T_A(x)^\gamma] = O(|x|)$.*

Although this definition looks peculiar, note that it is quite similar to the naive definition, which can be reformulated as asserting $\mathrm{E}[T_A(x)]^\gamma = O(|x|)$: the different is merely in the order of applying the expectation and powering operations. Definition 15 reflects a better understanding of the nature of the expectation operator (with respect to its interaction with other operations), and is preferable for the purpose of introducing a robust theory of efficient probabilistic algorithms. Indeed, Definition 15 implies the standard definition of expected PPT, but the fact that Definition 15 goes beyond expected PPT is of some concern in the current setting. Furthermore, keeping track of the actual expected running-time (as in the standard notion of expected PPT) seems better for the purpose of actually analyzing the running-time of simulators (especially, because our aim is comparing these to the running time of corresponding adversaries). For that reason, we performed our main treatment in terms of expected PPT, and only comment here on how it can be adapted to Definition 15.

Indeed, let us turn to our own business. For simplicity of exposition, note that Definition 15 remains intact if we require that $\mathrm{E}[T_A(x)^\gamma] = \mathrm{poly}(|x|)$ (rather than $\mathrm{E}[T_A(x)^\gamma] = O(|x|)$).[39] Next, note that the definitions presented in Section 2 can be adapted by merely replacing the random variable that represents the running-time (or the number of interactions) by its γ-th power, for some constant $\gamma > 0$. Let us demonstrate this adaptation for the most complicated case, where we use the related formulation of saying that the running-time is polynomial in a quantity that has polynomial expectation.

Definition 16 (Definition 3, revisited): *A q-reset attack on σ is an attack that, for every x, y, z and ω, interacts with σ_ω for an number of times that is upper-bounded by $q(X_{x,y,z,\omega})$ such that $\mathrm{E}[X_{x,y,z,\omega}] \leq \mathrm{poly}(|x|)$. The strategy σ is efficient w.r.t any reset attack if, for some polynomial p, every polynomial q, every q-reset attack on σ, and every x, y, z, the total number of steps taken by $\sigma(x, z)$ during this attack is upper-bounded by $q(Y_{x,y,z}) \cdot p(Y_{x,y,z})$ such that $\mathrm{E}[Y_{x,y,z}] \leq \mathrm{poly}(|x|)$.*

The analogues of Definitions 1, 2, and 4 are easier to obtain. We similarly adapt the definition of normal (black-box) simulator (i.e., Definition 6). It is left to verify that the analogous results remain valid.

Proposition 5, revisited. With the exception of the proof that Definition 4 implies Definition 3, all the claims are highly insensitive to the specific notion of efficient probabilistic computation. When proving the remaining analogue (and referring to notations as in the said proof), note that $\mathrm{E}_{r,\omega}[T_{A(r)}(\omega)^\gamma]$ is upper-bounded by the product of $\max_\omega\{\mathrm{E}_r[n_{A(r)}(\omega)^\gamma]\}$ and $\mathrm{E}_\omega[\max_r\{t_{A(r)}(\omega)^\gamma\}]$. Thus, upper-bounds on the latter factors[40] yield the desired upper-bound, which implies that any strategy that satisfies the analogue of Definition 4 also satisfies the analogue of Definition 3. ■

Theorem 8, revisited. Noting that Observation 7 extends to the current setting, we infer that so does the running-time analysis of the simulator. Thus, it remains to consider the analysis of the quality of the simulator's output. Let $T^*(x)$ be a random variable representing the total number of steps taken by V^* during all its invocations by M. (Recall that in the proof of Theorem 8 we only considered the expectation of this number, which was denoted $t^*(x)$ and indeed equals $\mathrm{E}[T^*(x)]$.) Then, for some $\gamma > 0$, it holds that $\overline{\mu} \stackrel{\mathrm{def}}{=} \mathrm{E}[(T^*(x))^\gamma] = \mathrm{poly}(|x|)$. Thus, $\Pr[(T^*(x))^\gamma > 3\overline{\mu}/\epsilon(|x|)] < \epsilon(|x|)/3$ and $\Pr[T^*(x) > (3\overline{\mu}/\epsilon)^{1/\gamma}] < \epsilon/3$ follows. Truncating runs of V^* once $(\mathrm{poly}(|x|)/\epsilon)^{1/\gamma}$ steps are completed, we obtain a strict PPT V^{**} and continue as in the original analysis. ■

On the extendability of Barak's non-black-box simulation. We claim that Barak's simulator [B01] (as well as its modification [BG08]) can handle adversaries that satisfy Definition 16 (or actually even a corresponding version of Definition 2). This claim can be proved by noting that this simulator can be applied to any deterministic adversary, while running in time that is polynomial in the running-time of the adversary. Thus, we may apply this simulator to a residual deterministic adversary obtained by fixing (at random) the coins of the given probabilistic adversary. It follows that the simulator makes a number of steps that satisfies Definition 15 (because its running-time is polynomial in a quantity that satisfies Definition 15). As for the quality of this simulation, it

[39]This follows by observing that, for every $c \geq 1$ and $X \geq 0$, it holds that $\mathrm{E}[X^c] \geq \mathrm{E}[X]^c$. Hence, $\mathrm{E}[T_A(x)^\gamma] = O(|x|)^c$ implies $\mathrm{E}[T_A(x)^{\gamma/c}] = O(|x|)$.

[40]Note that we may need to use the fact that $\mathrm{E}[X^\gamma] \leq \mathrm{E}[X^{\gamma'}]$ for every $\gamma \leq \gamma'$.

can be analyzed as the foregoing version of Theorem 8, while noting that the effect (on this specific simulator) of replacing of the code of V^* by the code of V^{**} is limited to the effect that this replacement has on a black-box simulator.[41]

Note that Theorem 9 as well as the other composition theorems are highly insensitive to the specific notion of efficient probabilistic computation in use. Their proof merely invokes the corresponding composition theorem for strict PPT and the relevant extendability theorem (e.g., Theorem 8). We thus conclude this section by considering the extendability theorem for general protocols.

Theorem 10, revisited. The issue again is the analysis of the running-time of the combined simulator (which in this context is an interactive machine). For the analogue of Definition 4, it suffices to relate to powers of the quantities appearing in the proof of Theorem 10 (rather than to the quantities themselves), indeed as done in the proof of the revisited Proposition 5. For the analogue of Definition 3 the modification is even more transparent. ■

5.2 Extending other approaches

We consider a relaxation of Definition 15 suggested by Vadhan [V06] as well as a generalization of the notion of epsilon-knowledge (used in, e.g., [DNS]).

5.2.1 Extending Vadhan's relaxation of Levin's approach

Let us start by noting that an equivalent formulation of Definition 15 asserts that, for some constant $\gamma > 0$, it holds that $\Pr[T_A(x) > t] = O(|x|/t^\gamma)$ for every x and t. Clearly, $\mathrm{E}[T_A(x)^\gamma] = O(|x|)$ implies $\Pr[T_A(x)^\gamma > t^\gamma] = O(|x|)/t^\gamma$ (for every t). On the other hand, if for some constant $\gamma > 0$ it holds that $\Pr[T_A(x)^\gamma > t^\gamma] = O(|x|)/t^\gamma$ (for every t), then, for every $\gamma' < \gamma$, it holds that $\mathrm{E}[T_A(x)^{\gamma'}] = O(|x|)$.

The foregoing equivalent form of Definition 15 is the starting point for further relaxation, suggested by Vadhan [V06]. According to this relaxation it is only required that *for some negligible function* $\mu : \{0,1\}^* \to [0,1]$ *it holds that* $\Pr[T_A(x) > t] = O(|x|/t^\gamma) + \mu(|x|)$. This relaxation is conceptually appealing, because a negligible deviation of various probabilities is allowed throughout the theory of cryptography.

Extending Vadhan's approach. Again, the definitional treatment provided in Section 2 and Definition 6 is easily adapted to the current notion of probabilistic efficiency. As for the analogous results, they all hold. This can be proved by noting that, for each relevant probabilistic process, all but a negligible measure of the probability space behaves analogously to Definition 15. Thus, the analysis used in Section 5.1 can be applied to the non-exceptional part of the probability space, and the negligible part can be ignored (or rather accounted for by the negligible error probability allowed in the final result). We stress that this decomposition of the probability space is only a mental experiment performed in the analysis, while the various strategies and algorithms remain exactly as described in Section 5.1.

5.2.2 Extending the epsilon-knowledge approach

Our starting point is an approach that was used in the context of zero-knowledge, where it is called *epsilon-knowledge*. In this approach the simulator is provided with a *non-negligible deviation*

[41] This is the case since this specific simulator only uses the code in order to generate a proof that the corresponding verifier behaves in a certain way.

parameter, denoted ϵ. The simulator is required to run in (strict) $\mathrm{poly}(|x|/\epsilon)$-time and should output a transcript that is ϵ-indistinguishable from the real interaction (i.e., the probability gap observed by any PPT distinguisher is at most ϵ (rather than negligible)). We stress that the running-time of the simulator may depend on ϵ. Intuitively, when the simulator is required to produced an output of higher quality (i.e., corresponding to a smaller ϵ), it is allowed more time.

From epsilon-knowledge to epsilon-security. Although (to the best of our knowledge) this approach has only been applied in the context of zero-knowledge, it can be applied to general secure protocol yielding a corresponding notion of *epsilon-security*.[42] That is, we consider arbitrary (non-interactive (and later interactive)) probabilistic machines that are given a parameter ϵ, and always run for at most $\mathrm{poly}(|x|/\epsilon)$ steps; both the adversary and its simulator will be modeled as such machines, but they may use different values of the deviation parameter (e.g., when given the parameter ϵ, the simulator may invoke the adversary with parameter ϵ'). Hence, epsilon-security means that for every such adversary there exists a corresponding simulator that, when given the parameter ϵ, yields an ideal execution that is ϵ-indistinguishable from the real one.

Note that in the foregoing paragraph we postulated that the relevant interactive (probabilistic) machine *always run for at most* $\mathrm{poly}(|x|/\epsilon)$ *steps*. That is, this formulation refers to strict running-time (and corresponds to strict PPT). We note that considering expected running-time (which corresponds to expected PPT) buys us nothing in the setting of epsilon-security, because one can always truncate runs that exceed the expected value by a factor of $1/\epsilon'$ (while incurring only a deviation of ϵ' in performance). Thus, the notion of epsilon-security is actually unrelated to the issues discussed in this paper.

On the composition of epsilon-secure protocols. We seize the opportunity to pointing out an important detail regarding the composition of epsilon-secure protocols. Suppose that we are composing an oracle-aided protocol Π with a protocol ρ, obtaining a protocol Π'. Recall that when constructing a simulator for Π', we use a simulator for Π that refers to an adversary A, where A itself incorporates a simulator for ρ, which is being invoked $t(|x|)$ times. Thus, the deviation of the combined simulator (for Π') is upper-bounded by $\epsilon + n_\epsilon(|x|) \cdot t(|x|) \cdot \epsilon_\rho$, where ϵ (resp., ϵ_ρ) is the deviation of the simulator of Π (resp., of ρ), and n_ϵ is an upper-bound on the number of invocations of A (by the simulator for Π). Note that n_ϵ is upper-bounded by the running-time of the simulator of Π, which in turn may depend on its own deviation parameter (i.e., ϵ); that is, we may have $n_\epsilon(|x|) = \mathrm{poly}(|x|/\epsilon)$. Thus, the total deviation of the combined simulator (for Π') may take the form $\epsilon + \mathrm{poly}(|x|/\epsilon) \cdot \epsilon_\rho$. This means that, when given a deviation parameter ϵ', the combined simulator should invoke the two simulators with sufficiently small deviation parameters (e.g., setting $\epsilon = \epsilon'/2$ and $\epsilon_\rho = \mathrm{poly}(\epsilon/|x|)$ will do).

6 Conclusions and Open Problems

We believe that the new definitions of expected PPT (i.e., Definitions 3 and 4) are satisfactory. Indeed, our belief is supported by the results presented in this paper; that is, by the fact that normal black-box simulators that handle strict PPT adversaries also handle adversaries that satisfy our definitions, and that these definitions support various natural composition theorems.

[42]In fact, a related notion of security has appeared in the context of password-based security (cf. [GL06]), but there ϵ is not a free parameter but rather represents the noticeable *a priori* probability of guessing the correct password, and the running-time of the simulator is independent of ϵ.

We note that both definitions arise naturally. As we saw, Definition 3 arises as the natural answer to the problem caused by dealing with adversaries that are expected PPT under Definition 2. As for Definition 4 it is simplest to state, and, contrary to our initial feeling, it works just as well. A natural question that arises is *which definition is preferable*: *Definition 3 or Definition 4*? At this point we feel no urge to address this question. In our opinion, a choice will have to be made only once we reach applications that work with one definition but not with the other.

We note that normal black-box simulators are pivotal to our main results. It may be that the same results (or equally satisfactory modifications of them) hold also for arbitrary black-box simulators and even for any universal simulators, but the current proofs fail to show this (see Footnote 26). We leave the resolution of this issue as an open problem. A good place to start may be getting rid of the normality condition.

Acknowledgments

I am grateful to Salil Vadhan for a discussion that inspired this work (and in particular Definition 3). I should be equally grateful to Yehuda Lindell for a discussion that inspired Definition 4, but I only understood this in retrospect. In addition, I wish to thank Salil and Yehuda for many insightful discussions and helpful comments on earlier drafts of this write-up. Finally, I wish to thank the reviewers of *TCC'07* and the *Journal of Cryptology* for their comments. Although I disagree with some of their conceptual comments, I found these comments interesting and challenging.

References

[B01] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[BG08] B. Barak and O. Goldreich, Universal arguments and their applications. *SIAM Journal on Computing*, Vol. 38, pages 1661–1694, 2008. Preliminary version in *17th CCC*, 2002.

[BL02] B. Barak and Y. Lindell. Strict Polynomial-time in Simulation and Extraction. In *34th ACM Symposium on the Theory of Computing*, pages 484–493, 2002.

[C00] R. Canetti. Security and Composition of Multi-party Cryptographic Protocols. *Journal of Cryptology*, Vol. 13, No. 1, pages 143–202, 2000.

[C01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001. Full version is available from the author.

[CGGM] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable Zero-Knowledge. In *32nd ACM Symposium on the Theory of Computing*, pages 235–244, 2000.

[DNS] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. In *30th ACM Symposium on the Theory of Computing*, pages 409–418, 1998.

[F90] U. Feige. *Alternative Models for Zero-Knowledge Interactive Proofs.* Ph.D Thesis, Weizmann Institute of Science, 1990.

[G97] O. Goldreich. Notes on Levin's Theory of Average-Case Complexity. *ECCC*, TR97-058, Dec. 1997.

[G01] O. Goldreich. *Foundation of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[G04] O. Goldreich. *Foundation of Cryptography: Basic Applications.* Cambridge University Press, 2004.

[GK96] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, Vol. 9, No. 2, pages 167–189, 1996. Preliminary versions date to 1988.

[GL06] O. Goldreich and Y. Lindell. Session-Key Generation using Human Passwords Only. *Journal of Cryptology*, Vol. 91, No. 3, pages 241–340, July 2006.

[GMW] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol. 38, No. 1, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

[GO94] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology*, Vol. 7, No. 1, pages 1–32, 1994.

[GMR] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing*, Vol. 18, pages 186–208, 1989. Preliminary version in *17th STOC*, 1985.

[KL08] J. Katz and Y. Lindell. Handling Expected Polynomial-Time Strategies in Simulation-Based Security Proofs. *Journal of Cryptology*, Vol. 21, pages 303–349, 2008. Preliminary version in *2nd TCC*, 2005.

[L86] L.A. Levin. Average Case Complete Problems. *SIAM Journal on Computing*, Vol. 15, pages 285–286, 1986.

[L09] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. *Journal of Cryptology*, Vol. 22, pages 395–428, 2009. Preliminary version in *44th FOCS*, 2003.

[MP06] S. Micali and R. Pass. Local Zero-Knowledge. In *38th ACM Symposium on the Theory of Computing*, pages 306–315, 2006.

[V06] S. Vadhan. Alternatives to expected probabilistic polynomial-time. Private communication, 2006.