On Expected Probabilistic Polynomial-Time Adversaries: A suggestion for restricted definitions and their benefits[*]

Oded Goldreich

Weizmann Institute of Science, Rehovot, ISRAEL

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.

1 An Opinionated Introduction

The title of this introduction and the use of first person singular 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 use of 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

[*] This research was partially supported by the Israel Science Foundation (grant No. 460/05).

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.

This is a trimmed version of my technical report [G06]. In particular, Sections 4-6 were omitted.

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 first, 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. After all, everybody seems quite happy with replacing one polynomial (bound

[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 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])).

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 there is something unpleasing about this discrepancy. Following [KL05], let me refer to this issue as an aesthetic consideration.

Jumping ahead in time, let me mention a more acute consideration articulated in [KL05]: 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 discussed 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. (Furthermore, Feige's definition also extends the standard definition of *strict* probabilistic polynomial-time adversaries by allowing adversaries that may not even halt when interacting with strategies other the those they were designed to attack (see proof of Proposition 5).)

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 [KL05], 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*

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

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]

Recently, Katz and Lindell [KL05] initiated a study of the possibility of simulating *expected* polynomial-time adversaries and/or obtaining composition theorems (or sufficiently good alternatives) for the *expected* polynomial-time case. 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 obtained 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

[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.

[4] Roughly speaking, the two main results of [KL05] 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 [KL05] 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.

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 unintuitive nature of expected running-time especially when applied in the context of cryptography[5] and by numerous annoying phenomena related to expected-time complexity. In particular, note 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.*[6] 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 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 25, which refers to a natural case in which this problem arises.)

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.[7]

[5] Indeed, things become even worse if we bear in mind the need to keep track of both the running-time and the success probability (which should be calculated with respect to various strict time bounds). That is, I claim that providing only the expected running-time and the overall success probability is quite meaningless, since the success is likely to be correlated with the running-time.

[6] See analogous discussion of average-case complexity in [G97].

[7] 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.

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.[8]
   (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. [KL05]), 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 being 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 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 [KL05] 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 forgoing 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.[9] (See Definition 3.)

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

[9] 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).

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 at the focus of the next subsection.

1.4 The main results

The main results establish the third desirable property for both definitions, while 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.[10]

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 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 polynomial in the length of the input, *where each oracle call is counted as a single step*, and call such a simulator normal. Indeed, the known (black-box) simulations including those that run in *expected* polynomial-time (e.g., [GK96]) are normal. For further discussion see the beginning of Section 3.

As stated in Section 1.3, the new definitions (or actually the "resetting-based" one) were devised to support the first main result (stated in [G06, Thm. 10]). This

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

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.[11]

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.[12]

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).[13] 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 adversary 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 [G06, Thm. 10] (or rather its zero-knowledge version – Theorem 8)).

[11] Note that Katz and Lindell [KL05] 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, [KL05, Sec. 3.3]).

[12] 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 itself to make sense at all it suffices to have a universal simulator (as otherwise it is not clear what we mean by saying that a simulator that handles any $A \in \mathcal{C}$ can handle any $A' \in \mathcal{C}'$).

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

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[14] 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 [G06, Thm. 10].

An important corollary to the foregoing extendability and composition theorems (i.e., [G06, Thm. 10] and [G06, Thm. 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

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

(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 [G06, Thm. 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 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*[15] refers to 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 this ratio that holds for every probabilistic polynomial-time adversary.[16] 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

[15] 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].

[16] Thus, if there exists a polynomial q such that, for every polynomial p, every p-time adversary is simulated in time $q \cdot p$ then the protocol has (noticeable) security tightness $1/q$. But if the simulation of p-time adversaries requires time p^3 then the protocol does not have a noticeable security tightness.

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*[17] 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]).

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., [G06, Thm. 10]) provided in this work as well as relevant composition theorems (e.g., [G06, Thm. 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).

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 measures 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 in my technical report [G06]. 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 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

[17] As usual, a noticeable function is one that decreases slower than the reciprocal of some positive polynomial.

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 of my technical report [G06]). Section 5 of [G06] discusses the applicability of my approach to alternative notions of expected polynomial-time algorithms, while Section 6 contains conclusions and open problems.

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.

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 [KL05]): *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).

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 polynomial.[18]

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 $q(|x|)$.*[19] *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 $q(|x|) \cdot p(|x|)$.*[20]

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 invocations of σ_ω is allowed to be an arbitrary random variable with a polynomially bounded expectation (rather than being strictly bounded by a polynomial).

[18] Indeed, the 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 initiated by the attacker (rather than its running-time), but do so not with respect to the designated σ but rather with respect to each of the residual σ_ω. 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.)

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

[20] 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.

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).

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 $p(|x|)$.*[21]

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 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 that satisfies Definition 4 *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 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 implication is provided in our technical report [G06].)

To show that the foregoing containments are strict we present corresponding strategies that witness the separations. The following examples are rather

[21] 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).

minimal, but they can be augmented into strategies that make sense (even for natural protocols). For example, a strategy that halts immediately upon receiving the message 0 and runs forever upon receiving the message 1 witnesses the separation between Definition 1 and Definition 2. 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: Consider a restriction of all four definitions such that each bound on an expectation is replaced by a corresponding strict bound. Then the resulting (strict) versions of Definition 2–4 coincide but remain separated from the (strict) version of Definition 1. We believe that this fact 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 our technical report [G06], analogous results are proved for general secure protocols.) To shorthand the text, when we say that some quantity (referring to an interaction) is polynomial, we mean that it is polynomial in the length of the common input.

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 strategy.[22] In extending this notion to randomized strategies, we refer to pro-

[22] In typical use of a black-box simulator one 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 not couple the operational aspect of the black-box simulator with the quality of the output that it produces, but rather separate the two.

viding 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. 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 make an* expected *number of steps that is upper-bounded by a polynomial in the length of the input,* where each oracle call is counted as a single step.

Although it is possible to construct black-box simulators that are not normal (e.g., they run forever 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 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 [KL05], 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.*

The straightforward proof is provided in our technical report [G06].

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.

[23] See corresponding footnote in our technical report [G06].

[24] The word “probably” indicates that the said implication is not claimed as a fact but rather suggested as a conjecture regarding any natural 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]).

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. ■

Discussion: 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.[25]

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 [G06, Sec. 4.1].

3.2 Sequential composition

The following 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 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.*

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 an 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, 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).

[25] 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 [BG02]) 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 $\Pr[t(x) = |x|^2] = 1 - 2^{-|x|}$.

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 forgoing, 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. Next, 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, 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. ■

Discussion: 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 [G06, Sec. 4.2].

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 for their comments, although I disagree with most comments.

References

[B01] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd FOCS*, pages 106–115, 2001.

[BG02] B. Barak and O. Goldreich, Universal arguments and their applications. In the *17th Conf. on Comput. Complex.*, pages 194–203, 2002.

[BL02] B. Barak and Y. Lindell. Strict Polynomial-time in Simulation and Extraction. In *34th STOC*, pages 484–493, 2002.

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

[C01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd FOCS*, 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 STOC*, pages 235–244, 2000.

[DNS] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. In *30th STOC*, 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.

[G06] O. Goldreich. On Expected Probabilistic Polynomial-Time Adversaries: A suggestion for restricted definitions and their benefits. *ECCC*, TR06-099, Aug. 2006. See revision, Nov. 2006.

[GK96] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. *J. of Crypto.*, 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. *J. of Crypto.*, 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. *JACM*, 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. *J. of Crypto.*, Vol. 7, No. 1, pages 1–32, 1994.

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

[KL05] J. Katz and Y. Lindell. Handling Expected Polynomial-Time Strategies in Simulation-Based Security Proofs. In *2nd TCC*, 2005. To appear in *J. of Crypto.*.

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

[L03] Y. Lindell. General Composition and Universal Composability in Secure Multi-Party Computation. In *44th FOCS*, pages 384–393, 2003.

[MP06] S. Micali and R. Pass. Local Zero-Knowledge. In *38th STOC*, 2006.