Handling Expected Polynomial-Time Strategies in Simulation-Based Security Proofs

Jonathan Katz[1] and Yehuda Lindell[2]*

[1] Department of Computer Science, University of Maryland, USA.
jkatz@cs.umd.edu
[2] Department of Computer Science, Bar-Ilan University, Israel.
lindell@cs.biu.ac.il

Abstract. The standard class of adversaries considered in cryptography is that of *strict* polynomial-time probabilistic machines (or circuits). However, *expected* polynomial-time machines are often also considered. For example, there are many zero-knowledge protocols for which the only simulation techniques known run in expected (and not strict) polynomial-time. In addition, it has been shown that expected polynomial-time simulation is *essential* for achieving constant-round black-box zero-knowledge protocols. This reliance on expected polynomial-time simulation introduces a number of conceptual and technical difficulties. In this paper, we develop techniques for dealing with expected polynomial-time adversaries in the context of simulation-based security proofs.

1 Introduction

Informally speaking, the simulation paradigm (introduced in [15]) states that a protocol is secure if the adversary's view in a real protocol execution can be generated solely from the information that it legitimately possesses (i.e., its input and output). The implication of this statement is that the adversary learns nothing from the protocol *execution*, since everything that the adversary sees in such an execution could be generated by the adversary itself. This paradigm can be instantiated in a number of different ways, where the differences that we refer to here relate to the complexity of the real adversary and the complexity of the simulator that generates the adversary's view.

The most straightforward way of instantiating the simulation paradigm is to require that for every strict polynomial-time adversary there exists a strict polynomial-time simulator that generates the required view. However, in many cases it is not known how to construct such simulators; rather, it is shown that for every *strict* polynomial-time adversary there exists an *expected* polynomial-time simulator that generates the required view. Essentially, this instantiation of the simulation paradigm has become the default one (at least for zero-knowledge). This reliance on expected polynomial-time simulation is problematic for the following reasons:

1. Aesthetic considerations: The intuition behind the simulation paradigm is that anything an adversary can learn from its interaction in a real protocol

* Most of this work was carried out while the author was at IBM T.J. Watson.

execution, it could also learn given only the input and the output. This follows because the adversary can run the simulator itself and thus obtain a view that is essentially the same as its view in a real execution. However, if the adversary is only allowed to run in strict polynomial-time while the simulator may run in expected polynomial-time, then the adversary *cannot* run the simulator (because it doesn't have enough time). One immediate solution to this problem is to allow the adversary to run in expected polynomial-time as well. However, as we will see in Section 1.1 below, this turns out to be problematic for technical reasons.

2. Technical considerations (composition): Consider the case that a secure protocol π calls a secure subprotocol ρ. Furthermore, both π and ρ are proven secure for strict polynomial-time adversaries using expected polynomial-time simulation. (Here, this means that π is proven secure under the assumption that ρ is replaced by some ideal function evaluation.) Now, the typical way of proving that π is secure when it calls the real subprotocol ρ is to first replace ρ with a simulated version, and then prove the security of π. However, this strategy will fail since it yields an *expected polynomial-time* adversary for π (because the adversary for π actually runs an internal expected polynomial-time simulation of ρ); yet π is proven secure only for strict polynomial-time adversaries.

In order to stress the implications of this difficulty, consider the following natural protocol. The parties first run a coin-tossing protocol (that uses expected polynomial-time simulation) in order to generate a common random string. Following this, the parties run a protocol that is secure in the common random string model (in this model, some trusted party provides both parties with the same uniformly distributed string). If the protocol that is designed for the common random string model is proven secure with respect to strict polynomial-time adversaries (which is usually the case), then the security of the coin-tossing protocol does not imply that the larger protocol is secure. The reason for this "gap" is the fact that simulation of the coin-tossing protocol yields an expected polynomial-time adversary, in the presence of which the protocol in the common random string model may not be secure. We remark that – seemingly due, at least in part, to these difficulties – all simulation-based composition theorems of which we are aware (e.g., [14, 4, 5]) deal only with the case of protocols proven secure via *strict* polynomial-time simulation.

In conclusion, expected polynomial-time simulation is currently a fact of life when it comes to proving the security of many cryptographic protocols. However, this causes difficulties especially when a protocol proven secure using expected polynomial-time simulation is used as a subprotocol.

1.1 Potential Ways of Resolving the Difficulties

There are at least two possible ways of dealing with the difficulties raised above:

1. Require simulators to be "as powerful" as adversaries: One way of resolving the above difficulties is to require simulators and adversaries to lie in the same complexity class. Here, there are two natural choices: **(a)** require both the adversary and the simulator to run in STRICT polynomial-time, or **(b)** allow both the adversary and the simulator to run in EXPECTED polynomial-time.

Limitations of the first choice (requiring STRICT polynomial-time for both adversary and simulator) were demonstrated in [3], who show that there do not exist *constant-round* zero-knowledge protocols with black-box simulators running in strict polynomial time. We note that non black-box simulation strategies running in strict polynomial-time are known to exist [1, 2]. However, all known "highly efficient" protocols are black-box. Thus, given our current knowledge, strict polynomial-time simulation techniques still pose a limitation on efficiency.

Before considering the second choice, where both simulators and adversaries run in EXPECTED polynomial-time, we briefly address the issue of defining expected polynomial-time adversaries. Loosely speaking, Feige [7] defined that an adversary $\mathcal{A}$ attacking a protocol π runs in expected polynomial-time if it runs in expected polynomial-time when interacting with the *honest parties running* π. Here, $\mathcal{A}$ may run for an unbounded amount of time when interacting with other machines (for example, an adversarial verifier for zero-knowledge needs only run in expected polynomial-time when interacting with the honest prover). The justification for such a definition is that the goal of an adversary is to attack an honest party. Therefore, any strategy that is "efficient" when interacting with an honest party is "feasible". We call this notion expected polynomial-time with respect to the protocol π. A more stringent definition, advocated by Goldreich [9], requires the adversary to run in expected polynomial-time when interacting with *any interactive machine*. We call this notion expected polynomial-time in any interaction. Clearly, any machine that is expected polynomial-time in any interaction is also expected polynomial-time with respect to any protocol π; it is also not hard to see that the converse is not true. Thus, the second notion defines a strictly smaller set of adversaries than the first.

We are now ready to discuss the implementation of the simulation paradigm in which both the adversary and the simulator run in expected polynomial-time. Feige [7] showed that the known simulation strategies for *computational* zero-knowledge all fail when considering adversaries that run in expected polynomial-time *with respect to the protocol*. In contrast, it was shown by [16, Appendix A.1] that the Feige-Shamir zero-knowledge argument system [7, 8] remains both zero-knowledge and an argument of knowledge even when the adversarial party runs in expected polynomial-time *in any interaction*. (We stress that the result of [16] does *not* hold for adversaries that run in expected polynomial-time with respect to the protocol.) It was further demonstrated by [16, Appendix A.2] that the known simulator for the Goldreich-Kahan zero-knowledge *proof* [1] system [11] does *not* remain zero-knowledge for adversaries that run in expected polynomial-time in any interaction (and so likewise for expected polynomial-time with respect to the protocol). Furthermore, there is no *computational proof system* that is known to remain zero-knowledge for adversaries that run in expected polynomial-time (under any definition). We therefore conclude that allowing

[1] Recall that in a proof system soundness holds even for all-powerful provers, whereas in an argument system it holds only for polynomial-time provers.

both the adversary and the simulator to run in EXPECTED polynomial-time is problematic *because we simply don't know how to construct simulators for such adversaries*. This is in contrast to the case when both the adversary and the simulator run in strict polynomial-time which, as we have mentioned, suffers from limitations which are inherent.

We remark that requiring simulators to be "as powerful" as adversaries addresses not only the aesthetic difficulty raised above, but also the issue of composition. This is due to the fact that once the simulator lies in the same class as the adversary, the general strategy for proving secure composition (as sketched above) is a viable one.

2. Prove a direct composition theorem: A second and incomparable approach addresses the technical issue of protocol composition, but does not deal with the above-mentioned aesthetic considerations. (Arguably, we can live more easily without aesthetics than without protocol composition.) In this approach, a composition theorem of the following type is proven: If two protocols π and ρ are both proven secure for strict polynomial-time adversaries while using expected polynomial-time simulation, then the composition of π with ρ is also secure for strict polynomial-time adversaries while using expected polynomial-time simulation. Such an approach may be pursued independently of the previous approach, and is worthwhile since many known protocols only satisfy the "strict/expected" notion of security. Namely, even if it is possible to construct protocols that are secure when both the adversary and the simulator run in expected polynomial-time, one may still want to use existing protocols that have been proven secure only for adversaries that run in strict polynomial-time (while using expected polynomial-time simulation).

1.2 Our Results

The main focus of this paper is to develop techniques for working with expected polynomial-time adversaries and simulation. We take the *first steps* in this direction and present two incomparable results, corresponding to the two approaches discussed in the previous section.

1. Simulation for expected polynomial-time adversaries. Our first result focuses on achieving expected polynomial-time simulation for expected polynomial-time adversaries. Before describing the result, we discuss one of the central technical problems that arises when dealing with expected polynomial-time adversaries: expected polynomial-time machines are not closed under "oracle composition". In more detail, let A be an oracle machine belonging to a class $\mathcal{C}$ and let B be any machine that also belongs to class $\mathcal{C}$. Then, we say the class $\mathcal{C}$ is closed under oracle composition if the machine A^B also belongs to $\mathcal{C}$ (when counting the steps of both A and B in their executions). This property of closure under oracle composition is important for black-box simulations (where machine A is the simulator and machine B is the adversary), and holds for the class of strict polynomial-time machines. However, the class of expected polynomial-time machines is *not* closed under oracle composition. To see this, consider the following two machines:

1. Machine A queries its oracle with the message 0 and receives back a message x. Next, A queries its oracle with x and halts.
2. Machine B receives an input q. If q equals its random tape r (where $|q| = |r| = k$, the security parameter), then B runs for 2^k steps and halts. Otherwise, it replies with r and halts.

Machine A runs in strict (and thus expected) polynomial-time. Likewise, machine B runs in expected polynomial-time because the probability (over choice of random tapes) that $q = r$ is 2^{-k} (and thus B runs for 2^k steps with probability 2^{-k}). However, the composed machine A^B always runs for more than 2^k steps. We therefore conclude that the composition of an expected polynomial-time simulator with an expected polynomial-time adversary may not yield an expected polynomial-time simulation. We stress that this problem is not just hypothetical. Rather, as we have mentioned earlier, many concrete protocols and expected polynomial-time simulators suffer from this problem [7, 16]. Furthermore, simple solutions, like truncating the execution after some polynomial number of steps, do not work; see [3] for some discussion.

Ideally, we would like to present conditions under which closure under oracle composition can be achieved for expected polynomial-time machines. This would allow us to construct an expected polynomial-time simulator that fulfills the conditions, and immediately derive simulation even when the adversary runs in expected polynomial-time. Toward this goal, we prove a theorem that shows how to automatically convert a class of simulators (characterized by a certain property) so that they remain expected polynomial-time even if the adversary runs in expected polynomial-time. More precisely, let $\mathcal{S}$ be a black-box simulator with the following two properties:

1. $\mathcal{S}$ runs in expected polynomial-time when given *any* oracle $\mathcal{A}$ (even if $\mathcal{A}$ is all-powerful). We stress that here we do not include $\mathcal{A}$'s running time in the complexity of $\mathcal{S}$. We also remark that most known black-box simulators have this property.
2. Every oracle query that $\mathcal{S}$ makes to its oracle $\mathcal{A}$ during its simulation is "strongly indistinguishable" to $\mathcal{A}$ from some partial view of a real protocol execution. By "strongly indistinguishable", we mean that the oracle query is computationally indistinguishable for circuits of size $\alpha(k)$, for some superpolynomial function $\alpha(k) = k^{\omega(1)}$. We remark that by making an appropriate $\alpha(k)$-hardness assumption, most known black-box simulators can be easily modified so that they fulfill this property.

Let $\mathcal{A}$ be an expected polynomial-time adversary and let $\mathcal{S}$ be a simulator that fulfills the above properties. We show that by truncating $\mathcal{S}^{\mathcal{A}}$ at $\alpha(k)$ steps, the resulting machine is a "good" simulator that runs in expected polynomial-time. We thus obtain a type of closure under oracle composition, as desired.

An important corollary of this theorem is a proof that, under mildly superpolynomial hardness assumptions, there exist computational zero-knowledge *proofs* for all $\mathcal{NP}$ that remain zero-knowledge even if the adversarial verifier runs in expected polynomial-time. As we have mentioned above, prior to this work no such proof system was known to exist. We note that our corollary has the following caveat: Our simulator for the zero-knowledge proof runs in expected polynomial-time only when given a statement x that is in the language L; see Section 3.3 for more details.[2]

We note that the above result does not achieve closure under oracle composition in its utmost generality, because it holds only for the above-described class of simulators. Nevertheless, many (if not most) known simulators can be modified so that they belong to this class. Furthermore, it is impossible to prove closure for *all* simulators, because closure under oracle composition for expected polynomial-time machines simply does not hold. Of course, it may still be possible to widen the class of simulators for which closure holds, and to remove the superpolynomial hardness assumptions.

2. A composition theorem. The above theorem holds for a restricted class of simulators, but achieves generality with respect to closure under oracle composition. Our second result is the opposite in that it holds for *all* black-box simulators, but relates only to a specific type of composition. Specifically, under a superpolynomial hardness assumption, we prove an analogue of the modular sequential composition theorem of Canetti [4] for protocols that are proven secure for strict polynomial-time adversaries using expected polynomial-time simulation. Loosely speaking, the modular sequential composition theorem of [4] states that if a secure protocol π contains *sequential* ideal calls to some functionalities, then it remains secure even when these ideal calls are replaced by *sequential* executions of subprotocols that securely realize the functionalities. The original result of [4] was previously known to hold only for protocols proven secure via *strict* polynomial-time simulation (in fact, in the full version we show that the proof of [4] fails in general for protocols proven secure via expected polynomial-time simulation). In contrast, our analogous result holds even if these protocols are proven secure using *expected* polynomial-time simulation (and only for *strict* polynomial-time adversaries). However, we also note that the proof of [4] requires no hardness assumptions, in contrast to ours which requires a superpolynomial hardness assumption.

We remark that both our results hold even for the larger class of adversaries running in expected polynomial-time with respect to the protocol under consideration [7].

Related work. The problem of simulation in expected polynomial-time was first posed by [7]; here we provide the first (partial) answers to some of the open questions posed there. The existence of constant-round zero-knowledge arguments with strict polynomial-time (non black-box) simulation was demonstrated

[2] Standard definitions require a simulator to generate a distribution that is indistinguishable from the view of the verifier *only* when it receives a statement $x \in L$. However, polynomial-time machines are typically required to run in polynomial-time for *all* inputs (i.e., even for $x \notin L$).

in [1, 2]. The feasibility of obtaining constant-round arguments *of knowledge* with strict polynomial-time extraction was then shown in [3]. They also showed that such protocols do not exist when the simulator or extractor is black-box. Thus, the protocols of [1–3] provide an alternative to expected polynomial-time simulation. In this work, we take a different approach and develop techniques for working with expected polynomial-time simulation. This has the advantage of not ruling out the many protocols (including most of the highly efficient protocols) that rely on expected polynomial-time simulation.

2 Definitions and Preliminaries

The security parameter is denoted by k; for conciseness, we equate the security parameter with the input length. (We therefore consider security for "sufficiently long inputs".) We denote by $A(x, z, r)$ the output of machine A on input x, auxiliary input z, and random coins r. The running time of A is measured in terms of the length of its first input x (where $|x| = k$), and the exact running time of the deterministic computation $A(x, z, r)$ is denoted by $\mathsf{time}_A(A(x, z, r))$. A runs in strict polynomial time if there is a polynomial $p(\cdot)$ such that for all x, z, and *all* r, it holds that $\mathsf{time}_A(A(x, z, r)) \leq p(|x|)$. A runs in expected polynomial time if there is a polynomial $p(\cdot)$ such that for all x and z, it holds that $\mathbf{Exp}_r[\mathsf{time}_A(A(x, z, r))] \leq p(|x|)$.

Running time for ITMs. If A is an interactive Turing machine (ITM), we let $A(x, z, r; \cdot)$ denote the "next message function" of A on inputs x, z, and random coins r. The ITM A runs in strict polynomial time if there is a polynomial $p(\cdot)$ such that for all x, z, r, and any sequence of messages $\overline{m}$, it holds that $\mathsf{time}_A(A(x, z, r; \overline{m})) \leq p(|x|)$.

Defining expected polynomial-time ITMs is more complicated, and at least two such definitions have been considered. We first present the definition of Feige [7]. As mentioned in the Introduction, the idea behind this definition is that any adversarial strategy that is efficient when run against the specified target is feasible. Thus, the running-time of an adversary when interacting with an arbitrary ITM (that is not the honest party under attack) is irrelevant. Informally, an ITM A is therefore said to run in expected polynomial-time with respect to a particular protocol π if there exists a polynomial $p(\cdot)$ such that for all inputs, the expected running time of A when interacting with *honest parties running* π is at most $p(|x|)$. (The expectation here is taken over the random coins of both A and the honest parties.) More formally, let $\mathsf{time}_A(\langle A(x, z_A, r), B(y, z_B, s)\rangle)$ denote the running time of A with input x, auxiliary input z_A, and random coins r, when interacting with B having input y, auxiliary input z_B, and random coins s. Then:

Definition 1 *An ITM A runs in* expected polynomial-time with respect to an ITM B *if there exists a polynomial $p(\cdot)$ such that for all x, y with $|x| = |y|$ and all auxiliary inputs $z_A, z_B \in \{0, 1\}^*$, the following holds:*

$$\mathbf{Exp}_{r,s}\left[\mathsf{time}_A(\langle A(x, z_A, r), B(y, z_B, s)\rangle)\right] \leq p(|x|).$$

Let $\pi = (P_1, P_2)$ be a two-party protocol. Then an adversary $\mathcal{A}$ runs in expected polynomial-time with respect to π *if it runs in expected polynomial-time with respect to P_1 and in expected polynomial-time with respect to P_2.*

The above definition relates to the case of two-party protocols. The extension to the multiparty case is obtained by considering the expected running-time of $\mathcal{A}$ when interacting (simultaneously) with every subset of honest parties.

As we have mentioned above, the fact that an adversary $\mathcal{A}$ runs in expected polynomial-time with respect to a protocol π means nothing about its running time when it interacts with other machines. A definition of the above sort makes sense in a cryptographic context, but is arguably a somewhat strange way of defining a "complexity class". An alternative approach advocated by Goldreich [9] therefore states that an ITM runs in expected polynomial time if there exists a polynomial $p(\cdot)$ such that for all inputs, the expected running time of A when interacting with *any* (even all powerful) ITM is at most $p(|x|)$. Here, the expectation is taken over the random coins of A only. In such a case, we say that A runs in expected polynomial-time in any interaction. More formally:

Definition 2 *An ITM $\mathcal{A}$ runs in* expected polynomial-time in any interaction *if for every ITM B it holds that A runs in expected polynomial-time with respect to B (as defined in Definition 1).*

It is immediate that if an ITM A runs in expected polynomial-time in any interaction, then A also runs in expected polynomial-time with respect to any protocol π. Furthermore, it is not difficult to show that for many protocols π, the class of adversaries running in expected polynomial-time with respect to π is *strictly larger* than the class of adversaries running in expected polynomial-time in any interaction. Since all our results hold even with respect to the stronger definition, and we view it as preferable in the cryptographic context, we adopt Definition 1 in this paper.

Expected polynomial-time oracle machines. Let A be an oracle machine that receives oracle access to an ITM B. In the execution of A with B, denoted by $A^{B(y,z_B,s;\cdot)}(x, z_A, r)$, machine A receives input x, auxiliary-input z_A and random tape r, and provides queries of the form $\overline{m}$ to its oracle which are answered as $B(y, z_B, s; \overline{m})$. We distinguish between two notions of running time for an oracle machine A^B:

1. $\mathsf{time}_A(A^{B(y,z_B,s;\cdot)}(x, z_A, r))$ denotes the exact running time of A on input x, auxiliary-input z_A, and random tape r when interacting with the oracle $B(y, z_B, s; \cdot)$, *counting calls to B as a single step* (i.e., we only "look" at the steps taken by A).
2. $\mathsf{time}_{A+B}(A^{B(y,z_B,s;\cdot)}(x, z_A, r))$ denotes the total running time of *both* A and B in the analogous execution. *Here, the steps taken by B to answer A's queries are also counted.*

Given the above, we can define expected polynomial-time oracle machines. An oracle machine A is said to run in expected polynomial-time if there exists a polynomial $p(\cdot)$ such that for every (even all powerful) machine B, all sufficiently-long inputs x, and every auxiliary input z, $\mathbf{Exp}_r[\mathsf{time}_A(A^B(x,z,r))] \leq p(|x|)$. Likewise, the composed machine A^B is said to run in expected polynomial-time if there exists a polynomial $p(\cdot)$ such that for all sufficiently-long inputs x and y with $|x| = |y|$, and all auxiliary inputs z_A and z_B, it holds that $\mathbf{Exp}_{r,s}[\mathsf{time}_{A+B}(A^{B(y,z_B,s)}(x,z_A,r))] \leq p(|x|)$. Note that for any strict polynomial-time B, if A runs in expected polynomial-time (not counting the steps of B) then so does A^B (where B's steps are counted). We stress, however, that this *does not necessarily hold* when B runs in expected polynomial time (under either definition considered earlier).

Requiring an expected polynomial-time oracle machine to run in the same (expected) amount of time when interacting with any machine B, even one which is computationally unbounded, seems to be overly stringent. However, all black-box simulators that we are aware of fulfill this condition. This extra condition is also *needed* for our results. We also remark that our definition of expected polynomial-time oracle and composed machines is *asymptotic*. That is, the machine is only required to run in (expected) time $p(|x|)$ for all long enough x's. As long as all machines considered halt on all inputs (and all random tapes), this is equivalent to the standard notion. (Indeed, we will assume this "halting condition" for all machines.)

3 Simulation for Expected Polynomial-Time Adversaries

In this section, we show how protocols proven secure against *strict* poly-time adversaries using a certain class of black-box simulation can in fact be proven secure against *expected* poly-time adversaries as well.

3.1 Preliminaries

As we have mentioned, the results of this section hold for a certain class of black-box simulators. We begin with a high-level description of secure computation, and then define the class of simulators. For the sake of simplicity, we present the results here for the case of two-party protocols. The extension to multiparty protocols is straightforward.

Secure two-party computation. We provide a very brief and informal overview of the definition of security for two-party computation. For more details, see [4, 10]. In the setting of two-party computation, two parties wish to jointly compute a (possibly probabilistic) functionality $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. That is, upon respective inputs x and y, the parties wish to compute $f(x,y)$ so that party P_1 receives $f_1(x,y)$ and party P_2 receives $f_2(x,y)$. Furthermore, the parties wish to ensure that nothing more than the output is revealed and that the function is correctly computed, even if one of the parties behaves adversarially. These requirements (and others) are formalized by comparing a real protocol execution to an ideal execution involving a trusted party. In an ideal execution with f, the parties send their inputs x and y to a trusted party who computes $f(x,y)$ and sends $f_1(x,y)$ to P_1 and $f_2(x,y)$ to P_2. Of course, the adversary who controls one of the parties can choose to send any input it wishes to the trusted party.[3] In contrast, in a real execution the parties P_1 and P_2 run a protocol π, where one of the parties may be corrupted and thus under the complete control of the adversary $\mathcal{A}$. Informally, we say a protocol π is secure if for every real-model adversary $\mathcal{A}$ interacting with an honest party running π, there exists an ideal-model adversary $\mathcal{S}$ interacting with a trusted party computing f, such that the output of $\mathcal{A}$ and the honest party in the real model is computationally indistinguishable from the output of $\mathcal{S}$ and the honest party in the ideal model. We note that in this work we consider *static adversaries* who corrupt one of the parties before the protocol execution begins.

Notation. Let $\pi = (P_1, P_2)$ be a two-party protocol and let f be a two-party functionality. We denote by $\text{REAL}_{\pi,\mathcal{A}}(x, y, z)$ the output of a *real execution of* π where party P_1 has input x, party P_2 has input y, and the adversary $\mathcal{A}$ has input z. Likewise, we denote by $\text{IDEAL}_{f,\mathcal{S}}(x, y, z)$ the output of an *ideal execution with* f where the respective inputs are as above. Since we are interested in black-box simulation, we present the definition for a black-box simulator $\mathcal{S}$:

Definition 3 (secure computation with black-box simulation): *Let f and π be as above. Protocol π is said to* black-box securely compute f (in the malicious model) *if there exists a non-uniform probabilistic expected polynomial-time oracle machine* (*ideal adversary/simulator*) $\mathcal{S}$ *such that for every non-uniform probabilistic polynomial-time real-model adversary $\mathcal{A}$, every non-uniform polynomial-time distinguisher D, every polynomial $p(\cdot)$, all sufficiently-long inputs x and y such that $|x| = |y|$, and all $z \in \{0,1\}^{\text{poly}(|x|)}$,*

$$\left|\Pr[D(\text{IDEAL}_{f,\mathcal{S}^{\mathcal{A}(z)}}(x, y, \lambda)) = 1] - \Pr[D(\text{REAL}_{\pi,\mathcal{A}}(x, y, z)) = 1]\right| < \frac{1}{p(|x|)}.$$

We note that $\mathcal{S}$ is an expected polynomial-time oracle machine as defined earlier. That is, for every $\mathcal{A}$ the expected value of $\mathsf{time}_{\mathcal{S}}(\mathcal{S}^{\mathcal{A}})$ is polynomial (even if $\mathcal{A}$ is computationally unbounded). To be more exact, however, the running-time of $\mathcal{S}$ may also depend on the messages it receives from the trusted party (and in particular, the random coins used by the trusted party to compute the functionality). We therefore denote by $\mathsf{time}_{\mathcal{S}}(\text{IDEAL}_{f,\mathcal{S}^{\mathcal{A}(z)}}(x, y, \lambda))$ the running-time of $\mathcal{S}^{\mathcal{A}}$ here. Adapting the earlier notation, we denote the expected running-time of $\mathcal{S}^{\mathcal{A}}$ *not counting* $\mathcal{A}$'s steps by $\mathbf{Exp}_s[\mathsf{time}_{\mathcal{S}}(\text{IDEAL}_{f,\mathcal{S}^{\mathcal{A}(z,r)}(1^{|z|},s)}(x, y, \lambda)]$, and its expected time *counting* $\mathcal{A}$'s steps by $\mathbf{Exp}_{r,s}[\mathsf{time}_{\mathcal{S}+\mathcal{A}}(\text{IDEAL}_{f,\mathcal{S}^{\mathcal{A}(z,r)}(1^{|z|},s)}(x, y, \lambda)]$. (The expectations above are actually also over the random-coins of the functionality. In this extended abstract, we ignore this issue.)

We now define a stronger notion of simulation which, informally, requires not only that the final output of $\text{IDEAL}_{f,\mathcal{S}^{\mathcal{A}}}$ be indistinguishable from $\text{REAL}_{\pi,\mathcal{A}}$, but also that each *partial* transcript generated during the simulation is indistinguishable from the (corresponding) partial transcript of a real execution of the protocol. Furthermore, we require that indistinguishability holds in a "strong" sense even against algorithms running in some slightly superpolynomial time. We begin by defining the following distributions:

[3] The adversary also has control over the delivery of the output from the trusted party to the honest party. Therefore, fairness and output delivery are not guaranteed.

1. $\text{SIM}_{f,\mathcal{S}^{\mathcal{A}}}(x, y, z, r, i)$ is defined by the following experiment: choose a random-tape $s \in_R \{0,1\}^*$ and run $\mathcal{S}^{\mathcal{A}(z,r;\cdot)}(1^{|z|}, s)$ in the ideal model with f. Let query_i be the i^{th} oracle query made by $\mathcal{S}$ to $\mathcal{A}$; if no such query is made, then set $\mathsf{query}_i = \bot$. Output query_i.
2. $\text{REAL}_{\pi,\mathcal{A}}(x, y, z, r, i)$ is defined by the following experiment: choose $s \in_R \{0,1\}^*$ and run a real execution where $\mathcal{A}$ has random-tape r and the honest party has random-tape s. Let $\overline{T}$ be the vector of messages sent by the honest party to $\mathcal{A}$ in this execution, and let $\overline{T}_j$ denote the first j messages in $\overline{T}$.
Next, run the experiment $\text{SIM}_{f,\mathcal{S}^{\mathcal{A}}}(x, y, z, r, i)$ above (with an independent choice of s) and obtain query_i. If $\mathsf{query}_i = \bot$, then output $\bot$. Otherwise, let j denote the number of messages in query_i, and output $\overline{T}_j$.

We note that the reason for running SIM in the second distribution is just to decide the length of the partial transcript to output. That is, we wish to compare the distribution of query_i to the partial transcript of a real execution *of the appropriate length*. We are now ready for the formal definition.

Definition 4 (α-strong black-box simulation): *Let π be a two-party protocol that is secure under black-box simulation, and let $\mathcal{S}$ be a black-box simulator for π. We say that $\mathcal{S}$ is an* α-strong black-box simulator for π (*and say that π is* secure under α-strong black-box simulation)*, if for every* strict polynomial-time *adversary $\mathcal{A}$, every non-uniform algorithm D running in time at most $\alpha(k)$, all $i \in \mathbf{N}$, all sufficiently large x and y, and all $z, r \in \{0,1\}^*$,*

$$\left|\Pr[D(\text{SIM}_{f,\mathcal{S}^{\mathcal{A}}}(x, y, z, r, i)) = 1] - \Pr[D(\text{REAL}_{\pi,\mathcal{A}}(x, y, z, r, i)) = 1]\right| < \frac{1}{\alpha(k)}.$$

If the above holds for adversaries $\mathcal{A}$ that are expected polynomial-time with respect to π, then we say that π is secure under α-strong black-box simulation for expected polynomial-time adversaries.

Extended black-box simulation. Finally, we introduce a generalization of black-box simulation in which the black-box simulator is allowed to *truncate* its oracle after it exceeds some (poly-time computable) number of steps $\alpha(\cdot)$. We call such a simulator extended black-box. We argue that this generalization is natural in the sense that the simulator still does not "look" at the internal workings of its oracle. We remark that when computing $\mathsf{time}_A(A^B)$, oracle calls are still considered a single step (even if A truncates B after some number of steps). Of course, $\mathsf{time}_{A+B}(A^B)$ also remains unchanged. We note that by requiring $\alpha(\cdot)$ to be polynomial-time computable, we ensure that any extended black-box simulator can be implemented by a non black-box simulator.

3.2 Simulation for Expected Polynomial-Time Adversaries

Theorem 5 *Let $\alpha(k) = k^{\omega(1)}$ be a superpolynomial function that is poly-time computable, and let π be a protocol that is secure under α-strong (extended) black-box simulation for* **strict** *polynomial-time adversaries. Then there exists a superpolynomial function $\alpha'(k)$ such that π is secure under α'-strong extended black-box simulation for* **expected** *polynomial-time adversaries.*

Proof: The idea behind the proof of this theorem is as follows. Since each query made by the α-strong simulator $\mathcal{S}$ to the real adversary $\mathcal{A}$ is indistinguishable from a partial real transcript *even for circuits of size* $\alpha(k)$, it follows that as long as $\mathcal{A}$ does not exceed $\alpha(k)$ steps, it cannot behave in a noticeably different way when receiving an oracle query or a real partial transcript. In particular, it cannot run longer when it receives an oracle query than it would run when interacting in a real protocol execution, and we know that it runs in expected polynomial-time in the latter case. We therefore construct a new simulator $\tilde{\mathcal{S}}$ that works in the same way as $\mathcal{S}$, except that it halts if $\mathcal{A}$ ever exceeds $O(\alpha(k))$ steps when answering a query. This enables us to prevent $\mathcal{A}$ from ever running for a very long time (something which can cause its expected running-time to be superpolynomial). Furthermore, by what we have claimed above, $\mathcal{A}$ will behave in almost the same way as before, because it can exceed $\alpha(k)$ steps only with probability that is inversely proportional to $\alpha(k)$. This will suffice for us to show that the new simulator is expected polynomial-time even if $\mathcal{A}$ is expected polynomial-time. Of course, we must also prove that the new simulation is no different than the old one. This follows again from the fact that $\mathcal{A}$ must behave in the "same way" as in a real execution, as long as $\alpha(k)$ steps are not exceeded. We now proceed with the actual proof.

Throughout the proof, we let k denote the length of x. Let $\mathcal{S}$ be the α-strong black-box simulator for π that is assumed to exist, and define $\hat{\mathcal{A}}$ as the algorithm that behaves exactly as $\mathcal{A}$ except that it outputs $\perp$ if it ever exceeds $\alpha(k)/2$ steps. Then, we construct a new simulator $\hat{\mathcal{S}}$ that receives oracle access to $\mathcal{A}$ and emulates a simulation of $\mathcal{S}$ with $\hat{\mathcal{A}}$. That is, $\hat{\mathcal{S}}$ chooses a random tape $s \in \{0,1\}^*$ and invokes $\mathcal{S}$ with random-tape s. Then, all oracle queries from $\mathcal{S}$ are forwarded by $\hat{\mathcal{S}}$ to its own oracle $\mathcal{A}$ and the oracle replies are returned to $\mathcal{S}$ unless the oracle exceeds $\alpha(k)/2$ steps while answering the query, in which case $\hat{\mathcal{S}}$ returns $\perp$ (thereby emulating $\hat{\mathcal{A}}$). Furthermore, all communication between $\mathcal{S}$ and the trusted party computing f is forwarded unmodified by $\hat{\mathcal{S}}$. We remark that $\hat{\mathcal{S}}$ is an *extended black-box simulator* because it truncates its oracle. (It makes no difference whether $\mathcal{S}$ was extended black-box or not.) We first show that $\hat{\mathcal{S}}$ runs in expected polynomial time, even when $\mathcal{A}$ runs in expected polynomial-time with respect to π.

Claim 6 *For every expected polynomial-time adversary $\mathcal{A}$, the composed machine $\hat{\mathcal{S}}^{\mathcal{A}}$ runs in expected polynomial time. That is, for every $\mathcal{A}$ there exists a polynomial $p(\cdot)$ such that all sufficiently large x and y, and all $z \in \{0,1\}^*$, it holds that* $\mathbf{Exp}_{r,s}[\mathsf{time}_{\hat{\mathcal{S}}+\mathcal{A}}(\mathrm{IDEAL}_{f,\hat{\mathcal{S}}^{\mathcal{A}(z,r)}(1^{|z|},s)}(x,y,\lambda)] \leq p(k)$.

Proof: To prove the claim, first note that the running time of $\hat{\mathcal{S}}$ consists of two components: the steps taken by $\mathcal{S}$ and the steps taken by $\hat{\mathcal{A}}$ in answering all of the oracle queries of $\mathcal{S}$. By the linearity of expectations, it suffices to show that the expectation of each of these components is polynomial. Since $\mathcal{S}$ is an expected polynomial-time oracle machine, its expected running time is polynomial when interacting with *any* oracle (see the end of Section 2). It therefore remains to bound the total number of steps taken by $\hat{\mathcal{A}}$. This is equal to

$\mathbf{Exp}_{r,s}[\sum_{i=1}^{\tau} \mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)]$, where τ is a random variable denoting the number of oracle queries made by $\mathcal{S}$, and $\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)$ is a random variable denoting the running time of $\hat{\mathcal{A}}(z,r)$ in answering the i^{th} query from $\mathcal{S}$. (Note that these random variables may depend on both r and s, and also on the honest party's inputs.) The expected value of τ is polynomial because $\mathcal{S}$ is an expected polynomial-time oracle machine. We now show that the expected value of $\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)$ is also polynomial for any i. Applying Wald's inequality (see Appendix A) then completes the proof that the expected total number of steps taken by $\hat{\mathcal{A}}$ is polynomial.

For any i, it holds that $\mathbf{Exp}_{r,s}[\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)] = \mathbf{Exp}_r[\mathbf{Exp}_s[\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)]]$. Furthermore, since $\hat{\mathcal{A}}$ halts after $\alpha(k)/2$ steps, it follows that for any *fixed* r,

$$\mathbf{Exp}_s\Big[\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)\Big] = \sum_{t=1}^{\alpha(k)/2} t\cdot\Pr_s\Big[\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i) = t\Big] = \sum_{t=1}^{\alpha(k)/2} \Pr_s\Big[\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i) \geq t\Big].$$

Notice that the distribution on the message sequence input to $\hat{\mathcal{A}}$ here (namely, the i^{th} query from $\mathcal{S}$) is exactly that given by $\textsc{sim}_{f,\mathcal{S}^{\hat{\mathcal{A}}}}(x,y,z,r,i)$. Now, let $\mathsf{time}_{\hat{\mathcal{A}}(z,r)}(i)$ be a random variable denoting the running time of $\hat{\mathcal{A}}(z,r)$ when run on input distributed according to $\textsc{real}_{\pi,\hat{\mathcal{A}}}(x,y,z,r,i)$. (Recall that this is a message of the same length as query_i, that $\hat{\mathcal{A}}$ receives in a real execution.) We first claim that, for large enough x and y, for any z,r,i, and for $t \leq \alpha(k)/2$,

$$\Big|\Pr_s[\mathsf{time}_{\hat{\mathcal{A}}(z,r)}(i) \geq t] - \Pr_s[\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i) \geq t]\Big| < \frac{1}{\alpha(k)}. \tag{1}$$

This follows because otherwise we obtain a non-uniform distinguisher, in contradiction to the fact that $\mathcal{S}$ is an α-strong black-box simulator. In more detail, given an auxiliary input $z' = (z,r,t)$ with $t \leq \alpha(k)/2$, and a sequence of j messages $\overline{T}_j$ we simply run $\hat{\mathcal{A}}(z,r)$ on message sequence $\overline{T}_j$, and output 1 iff $\hat{\mathcal{A}}$ exceeds t steps. For large enough k, the total running time of this distinguishing algorithm (including the overhead for maintaining a counter and running $\hat{\mathcal{A}}$) is at most $\alpha(k)$. Therefore, by Definition 4, it follows that Eq. (1) holds. We remark that the non-uniformity of Definition 4 is essential here. We thus have that:

$$\begin{aligned}\sum_{t=1}^{\alpha(k)/2} \Pr_s\left[\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i) \geq t\right] &\leq \sum_{t=1}^{\alpha(k)/2}\left(\Pr_s[\mathsf{time}_{\hat{\mathcal{A}}(z,r)}(i) \geq t] + \frac{1}{\alpha(k)}\right)\\ &= \frac{1}{2} + \sum_{t=1}^{\alpha(k)/2} \Pr_s[\mathsf{time}_{\hat{\mathcal{A}}(z,r)}(i) \geq t], \end{aligned} \tag{2}$$

and therefore the expected value of $\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)$ is bounded by the expression in Eq. (2). Using the simple observations that: **(1)** $\mathsf{time}_{\hat{\mathcal{A}}(z,r)}(i) \leq \mathsf{time}_{\hat{\mathcal{A}}(z,r)}$ (where the latter expression refers to the total running time of $\hat{\mathcal{A}}(z,r)$ in a real execution), and **(2)** $\mathsf{time}_{\hat{\mathcal{A}}(z,r)} \leq \mathsf{time}_{\mathcal{A}(z,r)}$ (because $\hat{\mathcal{A}}$ is truncated whereas $\mathcal{A}$ is not), we see that the expected value of $\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)$ is bounded by:

$$\frac{1}{2} + \sum_{t=1}^{\alpha(k)/2} \Pr_s[\mathsf{time}_{\mathcal{A}(z,r)} \geq t] \;\leq\; \frac{1}{2} + \mathbf{Exp}_s[\mathsf{time}_{\mathcal{A}(z,r)}]$$

where $\mathbf{Exp}_s[\mathsf{time}_{\mathcal{A}(z,r)}]$ is simply the expected running time of $\mathcal{A}$ in a real protocol execution with the *honest parties.* The fact that $\mathcal{A}$ runs in expected polynomial-time *with respect to* π therefore implies that the expected value of $\mathsf{time}^{\mathcal{S}}_{\hat{\mathcal{A}}(z,r)}(i)$ is polynomial, completing the proof of Claim 6. □

Until now, we have shown that $\hat{\mathcal{S}}$ runs in expected polynomial-time. It remains to show that it is an α'-strong (extended black-box) simulator for expected polynomial-time adversaries, for some superpolynomial function $\alpha'(k)$. First, $\hat{\mathcal{S}}$ is an expected polynomial-time oracle machine because it inherits this from $\mathcal{S}$. Next, we claim that for every expected polynomial-time $\mathcal{A}$, every non-uniform algorithm D running in time at most $\alpha(k)$, all $i \in \mathbf{N}$, all sufficiently large x and y, and all $z, r \in \{0,1\}^*$,

$$\left|\Pr[D(\textsc{sim}_{f,\hat{\mathcal{S}}^{\mathcal{A}}}(x,y,z,r,i)) = 1] - \Pr[D(\textsc{sim}_{f,\mathcal{S}^{\mathcal{A}}}(x,y,z,r,i)) = 1]\right| < \frac{1}{\alpha''(k)}$$

for some superpolynomial function $\alpha''(k)$. This follows from the facts that **(1)** the composed machine $\hat{\mathcal{S}}^{\mathcal{A}}$ runs in expected polynomial-time, and **(2)** the only time that $\hat{\mathcal{S}}^{\mathcal{A}}$ and $\mathcal{S}^{\mathcal{A}}$ differ is if $\mathcal{A}$ exceeds $\alpha(k)/2$ steps. That is, let $p(k)$ be the expected running time of the composed machine $\hat{\mathcal{S}}^{\mathcal{A}}$. Then, by Markov's inequality, the probability that $\hat{\mathcal{S}}^{\mathcal{A}}$ will exceed $\alpha(k)/2$ steps is at most $2p(k)/\alpha(k)$. Therefore, the statistical difference between $\textsc{sim}_{f,\hat{\mathcal{S}}^{\mathcal{A}}}(x,y,z,r,i)$ and $\textsc{sim}_{f,\mathcal{S}^{\mathcal{A}}}(x,y,z,r,i)$ is at most $\alpha''(k) \stackrel{\text{def}}{=} 2p(k)/\alpha(k)$. Combining this with the assumption that $\mathcal{S}$ is an α-strong simulator and so $\textsc{sim}_{f,\mathcal{S}^{\mathcal{A}}}(x,y,z,r,i)$ can be distinguished from $\textsc{real}_{\pi,\mathcal{A}}(x,y,z,r,i)$ with probability at most $1/\alpha(k)$, we conclude that

$$\left|\Pr[D(\textsc{sim}_{f,\hat{\mathcal{S}}^{\mathcal{A}}}(x,y,z,r,i)) = 1] - \Pr[D(\textsc{real}_{\pi,\mathcal{A}}(x,y,z,r,i)) = 1]\right| < \frac{1}{\alpha'(k)}$$

where $\alpha'(k) \stackrel{\text{def}}{=} (1/\alpha(k) + 1/\alpha''(k))^{-1}$. We conclude that $\hat{\mathcal{S}}$ is an α'-strong extended black-box simulator, as required. ■

3.3 Zero-Knowledge Proofs – A Corollary

Consider now the zero-knowledge functionality for an NP-language L. This function is defined by $f(x,x) = (\lambda, \chi_L(x))$, where $\chi_L(x) = 1$ if and only if $x \in L$. A zero-knowledge protocol is a protocol π that securely realizes f for strict polynomial-time adversaries. Now, for the sake of concreteness, consider the zero-knowledge protocol of Goldreich, Micali, and Wigderson [13]. Assuming the existence of commitment schemes that are hiding for circuits of size $\alpha(k)$, it is easy to verify that the black-box simulator provided by [13] is α-strong for strict polynomial-time adversaries. Therefore, by applying Theorem 5, we obtain that the protocol of [13] is also black-box secure for adversaries that run in expected polynomial-time with respect to the protocol. The soundness condition is unaffected by the above. We therefore obtain the *first* computational zero-knowledge *proof system* that remains zero-knowledge for expected polynomial-time adversaries (with respect to either of the definitions in Section 2).[4] Thus, as a corollary of Theorem 5, we partially resolve the open questions from [7, 16] discussed in the Introduction. (The result is only "partial" because we need superpolynomial hardness assumptions, and due to the caveat below.)

We remark that there is a subtle, yet important, caveat to the above. The simulator is only α-strong in the case that the input is a statement $x \in L$. This is due to the fact that when $x \notin L$, it may be possible for a distinguisher D to distinguish partial transcripts of the simulator from partial transcripts of a real execution just by checking if the statement is in the language (unless distinguishing $x \in L$ from $x \notin L$ is also assumed to be hard for circuits of size $\alpha(k)$). On the one hand, this is fine because simulators are only required to generate indistinguishable distributions in the case that $x \in L$. On the other hand, this is a problem because our simulator is not even guaranteed to *run* in expected polynomial-time for $x \notin L$. Thus, within a proof of security, one cannot invoke the zero-knowledge simulator on a statement x that may or may not be in the language, unless it is assumed that it is hard to distinguish $x \in L$ from $x \notin L$ in time $\alpha(k)$. In the full version of this paper, we discuss the ramifications of this caveat in greater detail.

3.4 Protocol Composition and Other Scenarios

We note that our result above has been stated for the stand-alone setting of secure computation. However, it actually holds for *any* setting, as long as the black-box simulator is α-strong for that setting. In particular, the result holds also for the setting of protocol composition where many protocol executions are run (and thus the simulator interacts with the trusted party many times).

4 A Modular Composition Theorem

Our goal in this section is to prove a modular composition theorem for secure multi-party computation which is analogous to the result of Canetti [4], but which holds even for protocols proven secure against *strict* polynomial-time adversaries while using *expected* polynomial-time simulation. As in Section 3, the results of this section are stated for the two-party case; the extension to the multiparty case is straightforward.

The sequential composition theorem of [4] can be informally described as follows. Let π be a two-party protocol computing a function g, designed in an (idealized) model in which the parties have access to a trusted party who evaluates functions $f_1, \ldots, f_m$; furthermore, assume that at most one ideal function call is made during any round of π. This model is called the $(f_1, \ldots, f_m)$-hybrid model, denoted $\textsc{hybrid}^{f_1,\ldots,f_m}$, because parties send real messages from the protocol π and also interact with a trusted party computing functions $f_1, \ldots, f_m$. Let $\rho_1, \ldots, \rho_m$ be a sequence of two-party protocols such that ρ_i securely computes f_i (as in Definition 3), and let $\pi^{\rho_1,\ldots,\rho_m}$ denote the "composed protocol" in

[4] In fact, computational zero-knowledge arguments were also not known to exist for adversaries that are expected polynomial-time with respect to the protocol.

which each ideal call to f_i is replaced by an invocation of ρ_i (we stress that each executed protocol ρ_i is run to completion before continuing the execution of π). The composition theorem then states that if π securely computes g in the hybrid model, and if each ρ_i securely computes f_i, then the composed real protocol $\pi^{\rho_1,\ldots,\rho_m}$ securely computes g. An important point to note is that the proof of [4] only considers the case that each of the component protocols ρ_i is proven secure via *strict* polynomial-time simulation. In fact, the *proof* of [4] demonstrably fails (in general) for the case of protocols proven secure via expected polynomial-time simulation; a counterexample is provided in the full version of this paper. In this section, we show that a suitable modification of the approach of [4] can be used to prove an analogous modular composition theorem even when each of the component protocols is proven secure via expected polynomial-time simulation.

We view this result as important both for conceptual reasons as well as for reasons of efficiency and practicality. Conceptually, there seems to be no fundamental reason that a composition theorem of this sort should not hold for the case of expected polynomial-time simulation; a number of technical barriers, however, make proving such a result difficult. From a practical point of view, many existing protocols – and, in particular, efficient ones – seem to require a proof of security via expected polynomial-time simulation. The composition theorem proven here enables protocol designers to enjoy the benefits of modular design and analysis, while ultimately allowing (more) efficient sub-protocols to be "plugged-in" for each of the components.

Preliminaries. We assume that the reader is familiar with [4], and so we borrow notation to the extent possible. In our proof, we use pseudorandom function families that are indistinguishable from random even for circuits of size $\alpha(k)$, for some superpolynomial function α. We call these α-secure pseudorandom functions.

The composition theorem. The composition theorem we prove is analogous to the one shown in [4] for the case of *strict* polynomial-time simulation. The only differences are that on the one hand, our proof holds also for the case of *expected* polynomial-time simulation, and on the other hand, we require *black-box* simulation and the existence of α-secure pseudorandom functions (the proof of [4] holds for any type of simulation and requires no hardness assumptions). We stress that, unlike in Section 3, here we consider the case that the real adversary runs in strict polynomial-time. Our proof of Theorem 7 is rather informal; a full and rigorous proof appears in the full version.

Theorem 7 *Assume the existence of $\alpha(k)$-secure pseudorandom functions for some $\alpha(k) = k^{\omega(1)}$. Let $f_1, \ldots, f_m$ and g be two-party functions, let π be an two-party protocol that black-box securely computes g in the $(f_1, \ldots, f_m)$-hybrid model where no more than one ideal evaluation call is made at each round, and let $\rho_1, \ldots, \rho_m$ be two-party protocols such that each ρ_i securely computes f_i. Then protocol $\pi^{\rho_1,\ldots,\rho_m}$ securely computes g.*

Proof: We follow the structure and notation of the proofs of [4, Theorems 5, 15] and [4, Corollaries 7, 17] as closely as possible. We focus on the case

$m = 1$; the general case follows easily using the techniques described here (and is omitted due to lack of space). We begin with a high-level overview of our proof, stressing where it diverges from [4]: Let $f = f_1$ be a two-party function, π a protocol in the f-hybrid model, ρ a protocol that securely computes f, and π^ρ the composed protocol. Given a strict polynomial-time adversary $\mathcal{A}$ in the real world (who interacts with parties running π^ρ), our goal is to construct an expected polynomial-time ideal-world adversary $\mathcal{S}$ (interacting with a trusted party who evaluates g) such that $\text{IDEAL}_{g,\mathcal{S}} \stackrel{c}{\equiv} \text{REAL}_{\pi^\rho,\mathcal{A}}$. We proceed in the following steps:

- As in [4], we first construct from $\mathcal{A}$ a (natural) real-world adversary $\mathcal{A}_\rho$ who interacts with parties running ρ as a stand-alone protocol. The security of ρ implies the existence of an expected polynomial-time simulator $\mathcal{S}_\rho$, who interacts with a trusted party evaluating f, such that $\text{IDEAL}_{f,\mathcal{S}_\rho} \stackrel{c}{\equiv} \text{REAL}_{\rho,\mathcal{A}_\rho}$.
- As in [4], using $\mathcal{A}$ and $\mathcal{S}_\rho$ we construct an adversary $\mathcal{A}_\pi$ interacting with parties running π in the f-hybrid model and satisfying $\text{HYBRID}^f_{\pi,\mathcal{A}_\pi} \stackrel{c}{\equiv} \text{REAL}_{\pi^\rho,\mathcal{A}}$. Contrary to [4], we *cannot* at this point claim the existence of an expected polynomial-time ideal-world adversary $\mathcal{S}$, who interacts with a trusted party evaluating g, such that $\text{IDEAL}_{g,\mathcal{S}} \stackrel{c}{\equiv} \text{HYBRID}^f_{\pi,\mathcal{A}_\pi}$ (such a claim, if true, would complete the proof). We cannot make such a claim because $\mathcal{A}_\pi$ runs in *expected* polynomial-time but the security of π only guarantees the existence of a "simulator" for *strict* polynomial-time adversaries.
- Instead, we first construct a modified adversary $\mathcal{A}'_\pi$ (still interacting with parties running π in the f-hybrid model) that runs in expected polynomial time and for which $\text{HYBRID}^f_{\pi,\mathcal{A}'_\pi} \stackrel{c}{\equiv} \text{HYBRID}^f_{\pi,\mathcal{A}_\pi}$ under the assumption that α-secure pseudorandom functions exist. This forms the crux of our proof, and further details are given below.
- Let $\mathcal{S}_\pi$ denote a black-box simulator for π (as in Definition 3). We define an ideal-world adversary $\mathcal{S}$ by running a slightly modified version of $\mathcal{S}_\pi$ with oracle access to $\mathcal{A}'_\pi$. We then prove that (1) $\text{IDEAL}_{g,\mathcal{S}} \stackrel{c}{\equiv} \text{HYBRID}^f_{\pi,\mathcal{A}'_\pi}$; and (2) that $\mathcal{S}$ runs in expected polynomial time (even when taking the running time of $\mathcal{A}'_\pi$ into account). The proof of the second claim relies on the existence of α-secure pseudorandom functions. We stress that we do not claim the above is true when $\mathcal{S}_\pi$ is run with oracle access to an *arbitrary* expected polynomial time machine (indeed, the claims may not be true if $\mathcal{S}_\pi$ is run with oracle access to the original $\mathcal{A}_\pi$), but rather we only make these claims with regard to the *specific* $\mathcal{A}'_\pi$ that we construct.

We now proceed with the proof. Since the first steps of our proof – namely, the construction of $\mathcal{A}_\rho$, $\mathcal{S}_\rho$, and $\mathcal{A}_\pi$ – are exactly as in [4], we omit the details here but instead provide only a high-level description of the adversary $\mathcal{A}_\pi$ which runs in the f-hybrid model. Loosely speaking, $\mathcal{A}_\pi$ runs $\mathcal{A}$ until the protocol ρ is supposed to begin. At this point, $\mathcal{A}$ expects to run ρ, whereas $\mathcal{A}_\pi$ should use an ideal call to f. Therefore, $\mathcal{A}_\pi$ invokes $\mathcal{S}_\rho$ giving it the current internal state z^ρ of $\mathcal{A}$ as its auxiliary input, and forwarding the messages between $\mathcal{S}_\rho$ and the trusted party computing f. The output of $\mathcal{S}_\rho$ is an internal state of $\mathcal{A}$ at the end of the execution of ρ; adversary $\mathcal{A}_\pi$ continues by invoking $\mathcal{A}$ on this state and running $\mathcal{A}$ until the conclusion of π. We remark that $\mathcal{A}_\pi$'s random-tape is parsed into r and r^*, and $\mathcal{A}_\pi$ invokes $\mathcal{A}$ with random-tape r and $\mathcal{S}_\rho$ with random-tape r^*. This concludes the (informal) description of $\mathcal{A}_\pi$. As in [4], it holds that $\text{HYBRID}^f_{\pi,\mathcal{A}_\pi} \stackrel{c}{\equiv} \text{REAL}_{\pi^\rho,\mathcal{A}}$. In this case, however, $\mathcal{A}_\pi$ is an *expected* polynomial-time adversary.

Sidetrack – motivation for the proof. At this point, it is possible to provide the key idea behind the proof of the theorem. Let $\mathcal{S}_\pi$ be the simulator that is guaranteed to exist by the fact that π black-box securely computes g in the f-hybrid model. Then, the main problem that arises in the proof of [4] is that the expected running-time of $\mathcal{S}_\pi$ when given access to the oracle $\mathcal{A}_\pi$ may not be polynomial. Consider the case that the strategy of $\mathcal{S}_\pi$ involves "rewinding" $\mathcal{A}_\pi$. Then, it is possible that $\mathcal{A}_\pi$ will invoke $\mathcal{S}_\rho$ *a number of times with the same random-tape* r^*. This introduces dependence between the executions, and may cause $\mathcal{S}_\rho$ to always run for a very long time. (The composition of the machines A and B described in the Introduction yielded an exponential-time machine exactly due to the fact that A invoked B with the same random-tape twice.) The first solution that comes to mind would be to have $\mathcal{A}_\pi$ choose an independent random-tape every time that it invokes $\mathcal{S}_\rho$. However, $\mathcal{S}_\pi$ works when given an oracle $\mathcal{A}_\pi$ with a *fixed* random-tape, and therefore this solution does not work. Our solution is to instead modify $\mathcal{A}_\pi$ so that it invokes $\mathcal{S}_\rho$ with a new pseudorandom tape each time (in a way reminiscent of a similar technique used in [6]). By using α-strong pseudorandom functions, we ensure that the pseudorandom tapes "look random" throughout the entire simulation by $\mathcal{S}_\pi$.

Back to the proof. As described above, we modify $\mathcal{A}_\pi$ to an adversary $\mathcal{A}'_\pi$, using a family $\mathcal{F}$ of α-secure pseudorandom functions for $\alpha(k) = k^{\omega(1)}$. The random tape of $\mathcal{A}'_\pi$ is parsed as r, s, where r is used exactly as above (i.e., $\mathcal{A}'_\pi$ invokes $\mathcal{A}$ with random-tape r), and s is used as a key to an α-secure pseudorandom function. Then $\mathcal{A}'_\pi$ sets the random-tape r^* for $\mathcal{S}_\rho$ to $r^* = F_s(z^\rho)$, where z_ρ is the current internal state of $\mathcal{A}$ when $\mathcal{S}_\rho$ is invoked (instead of choosing it randomly like $\mathcal{A}_\pi$). In addition, $\mathcal{A}'_\pi$ halts with output $\perp$ if it ever exceeds $\alpha(k)/2$ steps overall (not including steps used in computing F_s).[5] Apart from the above, $\mathcal{A}'_\pi$ works in exactly the same way as $\mathcal{A}_\pi$. We now prove the following claims:

Claim 8 *Assuming that $\mathcal{F}$ is an α-secure family of pseudorandom functions, $\mathcal{A}'_\pi$ runs in expected polynomial time.*

Proof (sketch): Consider a modified simulator $\hat{\mathcal{A}}_\pi$ who chooses a truly random-tape r^* for $\mathcal{S}_\rho$ instead of a pseudorandom one. (In particular, the only difference

[5] There is an additional subtlety here, in that $\mathcal{S}_\rho$ may require a superpolynomial number of coins while the output of F_s is polynomial. However, this can be easily resolved: by construction, we never require more than $\alpha(k)$ coins for $\mathcal{S}_\rho$.Coins for $\mathcal{S}_\rho$ can thus be generated as needed by letting the i^{th} coin required by $\mathcal{S}_\rho$ be given by $F_s(z^\rho | \langle i \rangle)$ where $\langle i \rangle$ is the $\log(\alpha(k))$-bit representation of i.

between $\hat{\mathcal{A}}_\pi$ and $\mathcal{A}_\pi$ is that $\hat{\mathcal{A}}_\pi$ outputs $\perp$ if it ever exceeds $\alpha(k)/2$ steps.) Then, the expected running time of $\hat{\mathcal{A}}_\pi$ on any set of global inputs global (which includes both the inputs explicitly given to $\hat{\mathcal{A}}_\pi$ as well as the inputs and random coins of the honest parties and the random coins of the trusted party) is at most:

$$\begin{aligned}\sum_{t=1}^{\alpha(k)/2} \Pr_{r,r^*}[\mathsf{time}_{\hat{\mathcal{A}}_\pi}(\mathsf{global}) \geq t] &\leq \sum_{t=1}^{\alpha(k)/2} \Pr_{r,r^*}[\mathsf{time}_{\mathcal{A}_\pi}(\mathsf{global}) \geq t] \\ &\leq \sum_{t=1}^{\infty} \Pr_{r,r^*}[\mathsf{time}_{\mathcal{A}_\pi}(\mathsf{global}) \geq t] \leq p_{\mathcal{A}_\pi}(k)\end{aligned}$$

where $p_{\mathcal{A}_\pi}(\cdot)$ is the polynomial upper-bound on the expected running-time of $\mathcal{A}_\pi$, and where we ignore the time required to maintain a counter for the number of steps (since this only affects the expected running time by a multiplicative polynomial factor). Now, since r^* is actually chosen pseudorandomly by $\mathcal{A}'_\pi$, we have that for large enough k, every value of global and all $t \leq \alpha(k)/2$:

$$\left|\Pr_{r,s}[\mathsf{time}_{\mathcal{A}'_\pi}(\mathsf{global}) \geq t] - \Pr_{r,r^*}[\mathsf{time}_{\hat{\mathcal{A}}_\pi}(\mathsf{global}) \geq t]\right| \leq \frac{1}{\alpha(k)}. \qquad (3)$$

(Eq. (3) ignores the time spent by $\mathcal{A}'_\pi$ in computing F_s because, as above, it only affects the expected running-time by a multiplicative polynomial factor.) Otherwise, we can construct a distinguisher D for $\mathcal{F}$ as in the proof of Claim 6 (details appear in the full version). We conclude that the expected running-time of $\mathcal{A}'_\pi$ on global inputs global and large enough k equals

$$\sum_{t=1}^{\alpha(k)/2} \Pr_{r,s}[\mathsf{time}_{\mathcal{A}'_\pi}(\mathsf{global}) \geq t] \leq \sum_{t=1}^{\alpha(k)/2} \left(\Pr_{r,r^*}[\mathsf{time}_{\hat{\mathcal{A}}_\pi}(\mathsf{global}) \geq t] + \frac{1}{\alpha(k)}\right)$$

which equals at most $p_{\mathcal{A}_\pi}(k) + 1$, and so is polynomial. □

Claim 9 *Assuming that $\mathcal{F}$ is an α-secure family of pseudorandom functions, it holds that* $\text{HYBRID}^f_{\pi,\mathcal{A}'_\pi} \stackrel{c}{\equiv} \text{HYBRID}^f_{\pi,\mathcal{A}_\pi}$.

Proof (sketch): Let $\hat{\mathcal{A}}_\pi$ be the same as in Claim 8. Since the expected running-time of $\mathcal{A}_\pi$ on security parameter 1^k is polynomial (for any set of global inputs), the probability that $\mathcal{A}_\pi$ exceeds $\alpha(k)/2$ steps is negligible. Hence $\text{HYBRID}^f_{\pi,\hat{\mathcal{A}}_\pi}$ is *statistically* close to $\text{HYBRID}^f_{\pi,\mathcal{A}_\pi}$. Now, $\mathcal{A}'_\pi$ is identical to $\hat{\mathcal{A}}_\pi$ except that it uses a pseudorandom r^* while $\hat{\mathcal{A}}_\pi$ uses a truly random r^*. Since $\mathcal{A}'_\pi$ and $\hat{\mathcal{A}}_\pi$ both run in at most $\alpha(k)/2$ steps (for the case of $\mathcal{A}'_\pi$, not counting the time required to compute F_s), the assumption that $\mathcal{F}$ is an α-secure family of pseudorandom functions immediately implies that $\text{HYBRID}^f_{\pi,\mathcal{A}'_\pi}$ is computationally indistinguishable from $\text{HYBRID}^f_{\pi,\hat{\mathcal{A}}_\pi}$ (details omitted), completing the proof. □

Defining the simulator $\mathcal{S}$. Since π *black-box* securely computes g, there exists an oracle machine $\mathcal{S}_\pi$ satisfying the conditions of Definition 3 (with appropriate modifications for comparing the f-hybrid and ideal models). Our simulator $\mathcal{S}$

works by simply invoking $\mathcal{S}_\pi$ with oracle $\mathcal{A}'_\pi$, with the limitation that it halts with output $\perp$ if it ever exceeds $\alpha(k)/2$ steps (including the running time of $\mathcal{A}'_\pi$ but, again, not including time spent computing F_s). Our aim is to show that **(1)** $\mathcal{S}$ runs in expected polynomial-time (even when taking the running time of $\mathcal{A}'_\pi$ into account), and **(2)** $\text{HYBRID}^f_{\pi,\mathcal{A}'_\pi} \stackrel{c}{\equiv} \text{IDEAL}_{g,\mathcal{S}}$. We stress that neither of these claims are immediate since $\mathcal{A}'_\pi$ is an *expected* polynomial-time adversary, and the simulator $\mathcal{S}_\pi$ has only been proven for the case that it is given a *strict* polynomial-time oracle.

Claim 10 *Assuming that $\mathcal{F}$ is an α-secure family of pseudorandom functions, $\mathcal{S}$ runs in expected polynomial time.*

Proof (sketch): We use the same general technique as in the proof of Claim 8, but the proof here is slightly more complicated. First imagine an adversary $\widetilde{\mathcal{S}}$ that differs from $\mathcal{S}$ in the following way: whenever $\mathcal{S}_\rho$ is called from within $\mathcal{A}'_\pi$, $\widetilde{\mathcal{S}}$ monitors the value of z^ρ at that point. Let z^ρ_i denote the i^{th} value of z^ρ in the execution of $\widetilde{\mathcal{S}}$. Then instead of setting $r^*_i = F_s(z^\rho_i)$, $\widetilde{\mathcal{S}}$ instead chooses r^*_i as follows: if $z^\rho_i = z^\rho_j$ for some $j < i$, then set $r^*_i = r^*_j$. Otherwise, choose r^*_i uniformly at random (the technicalities raised in footnote 5 can be handled in the obvious way). We first show that $\widetilde{\mathcal{S}}$ runs in expected polynomial-time, and then claim (as in the proof of Claim 8) that the expected running-times of $\widetilde{\mathcal{S}}$ and $\mathcal{S}$ cannot differ "too much".

The running time of $\widetilde{\mathcal{S}}$ is the sum of three components: $\mathsf{time}_{\mathcal{S}_\pi}$, the running time of $\mathcal{S}_\pi$ when counting its oracle calls to $\mathcal{A}'_\pi$ as a single step; $\mathsf{time}_{\mathcal{A}'_\pi}$, the running time of $\mathcal{A}'_\pi$ (when answering oracle calls of $\mathcal{S}_\pi$) but excluding time spent running $\mathcal{S}_\rho$; and $\mathsf{time}_{\mathcal{S}_\rho}$, and the running time of $\mathcal{S}_\rho$ when called by $\mathcal{A}'_\pi$ (each time $\mathcal{A}'_\pi$ is run).[6] By linearity of expectation, we can analyze each of these individually. The expected value of $\mathsf{time}_{\mathcal{S}_\pi}$ is polynomial since $\mathcal{S}_\pi$ is an expected polynomial-time oracle machine (as defined in Section 2). Furthermore, since $\mathcal{A}'_\pi$ runs in *strict* polynomial time when excluding the steps of $\mathcal{S}_\rho$, and since $\mathcal{S}_\pi$ makes an expected polynomial number of calls to $\mathcal{A}'_\pi$, the expected value of $\mathsf{time}_{\mathcal{A}'_\pi}$ is polynomial as well. It remains to analyze $\mathsf{time}_{\mathcal{S}_\rho}$. This variable is equal to $\sum_{i=1}^{\mathsf{time}_{\mathcal{S}_\pi}} \mathsf{time}_{\mathcal{S}_\rho}(i)$, where $\mathsf{time}_{\mathcal{S}_\rho}(i)$ represents the running time of $\mathcal{S}_\rho$ in its i^{th} execution. Since the random coins r^*_i used in the i^{th} execution of $\mathcal{S}_\rho$ are chosen at random, the expectation of $\mathsf{time}_{\mathcal{S}_\rho}(i)$ is polynomial for all i. Wald's inequality (cf. Appendix A) thus implies that the expected value of $\mathsf{time}_{\mathcal{S}_\rho}$ is polynomial.

Exactly as in the proof of Claim 8, the fact that $\mathcal{F}$ is α-secure can be used to show that $\mathcal{S}$ runs in expected polynomial time as well. We omit the details (which are identical) here. □

To complete the proof of the main theorem, we need to prove that $\text{IDEAL}_{g,\mathcal{S}} \stackrel{c}{\equiv} \text{HYBRID}^f_{\pi,\mathcal{A}'_\pi}$. The proof of this is largely similar to the end of the proof of Theorem 5 and appears in the full version of this paper. ■

[6] As discussed earlier, we again ignore time spent computing F_s.

References

1. B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *42nd FOCS*, pages 106–115, 2001.
2. B. Barak and O. Goldreich. Universal Arguments and their Applications. *17th IEEE Conference on Computational Complexity*, pages 194–203, 2002.
3. B. Barak and Y. Lindell. Strict Polynomial-Time in Simulation and Extraction. *SIAM Journal on Computing,* 33(4):783–818, 2004.
4. R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
5. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd FOCS*, pages 136–145, 2001.
6. R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable Zero-Knowledge. STOC 2000.
7. U. Feige. *Alternative Models for Zero Knowledge Interactive Proofs.* Ph.D. Thesis, Weizmann Institute, 1990.
8. U. Feige and A. Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *CRYPTO'89*, Springer-Verlag (LNCS 435), pages 526–544, 1989.
9. O. Goldreich. *Foundations of Cryptography: Volume 1 – Basic Tools.* Cambridge University Press, 2001.
10. O. Goldreich. *Foundations of Cryptography: Volume 2 – Basic Applications.* Cambridge University Press, 2004.
11. O. Goldreich and A. Kahan. How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.
12. O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing* 25(1):169–192, 1996.
13. 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* 38(1):691–729, 1991.
14. O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. *Journal of Cryptology* 7(1):1–32, 1994.
15. S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM Journal on Computing,* 18(1):186–208, 1989.
16. Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. *Journal of Cryptology*, 16(3):143–184, 2003.

A Wald's Inequality

We state a (slightly modified version of) Wald's inequality here. The proof is provided in the full version.

Lemma 11 *Let $Y_1, Y_2, \ldots$ be an infinite sequence of non-negative random variables such that $\mathbf{Exp}[Y_i] \leq N$ for all i. Let τ be a non-negative integer random variable for which, for all i, $\Pr[\tau = i]$ depends only on $Y_1, \ldots, Y_i$. Define $\overline{Y} \stackrel{\text{def}}{=} \sum_{i=1}^{\tau} Y_i$ (with the sum defined as 0 in case $\tau = 0$). Then $\mathbf{Exp}\left[\overline{Y}\right] \leq N \cdot \mathbf{Exp}[\tau]$.*