# Zero-Knowledge twenty years after its invention

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

PRELIMINARY DRAFT (July 31, 2002)

**Abstract**

Zero-knowledge proofs are proofs that are both convincing and yet yield nothing beyond the validity of the assertion being proven. Since their introduction about twenty years ago, zero-knowledge proofs have attracted a lot of attention and have, in turn, contributed to the development of other areas of cryptography and complexity theory.

We survey the main definitions and results regarding zero-knowledge proofs. Specifically, we present the basic definitional approach and its variants, results regarding the power of zero-knowledge proofs as well as recent results regarding questions such as the composability of zero-knowledge proofs and the use of the adversary's program within the proof of security (i.e., non-black-box simulation).

# Contents

# 1 Introduction

Zero-Knowledge proofs, introduced by Goldwasser, Micali and Rackoff [66], are fascinating and extremely useful constructs. Their fascinating nature is due to their seemingly contradictory definition; zero-knowledge proofs are both convincing and yet yield nothing beyond the validity of the assertion being proven. Their applicability in the domain of cryptography is vast; they are typically used to force malicious parties to behave according to a predetermined protocol. In addition to their direct applicability in Cryptography, zero-knowledge proofs serve as a good bench-mark for the study of various problems regarding cryptographic protocols (e.g., the "preservation of security under various forms of protocol composition" and the "use of of the adversary's program within the proof of security").

In this tutorial we present the basic definitions and results regarding zero-knowledge as well as some recent developments regarding this notion. The rest of the introduction provides a high-level summary of the tutorial.

## 1.1 The Basics

Loosely speaking, zero-knowledge proofs are proofs that yield nothing beyond the validity of the assertion. That is, a verifier obtaining such a proof only gains conviction in the validity of the assertion. This is formulated by saying that anything that is feasibly computable from a zero-knowledge proof is also feasibly computable from the (valid) assertion itself (by a so-called simulator). Variants on the basic definition include:

- Consideration of auxiliary inputs.
- Mandating of universal and black-box simulations.
- Restricting attention to honest (or rather semi-honest) verifiers.
- The level of similarity required of the simulation.

It is well-known that *zero-knowledge proofs exist for any NP-set*, provided that one-way functions exist. This result is a powerful tool in the design of cryptographic protocols, because it enables to force parties to behave according to a predetermined protocol (i.e., the protocol requires parties to provide zero-knowledge proofs of the correctness of their secret-based actions, without revealing these secrets).

**Organization of Part 1:** We start with some preliminaries (Section 2), which are central to the "mind-set" of the notion of zero-knowledge. In particular, we review the definitions of interactive proofs and arguments as well as the definitions of computational indistinguishability (which underlies the definition of general zero-knowledge) and of one-way functions (which are used in constructions). We then turn to the definitional treatment of zero-knowledge itself (Section 3). Finally, we discuss the constructibility and applicability of zero-knowledge proofs (Section 4).

## 1.2 Advanced Topics

We start with two basic problems regarding zero-knowledge, which actually arise also with respect to the security of other cryptographic primitives. The first question refers to the preservation of security (i.e., zero-knowledge in our case) under various types of composition operations. We survey the known results regarding sequential, parallel and concurrent execution of (arbitrary and/or specific) zero-knowledge protocols. The main facts are:

- Zero-knowledge (with respect to auxiliary inputs) is closed under sequential composition.
- In general, zero-knowledge is not closed under parallel composition. Yet, some zero-knowledge proofs (for NP) preserve their security when many copies are executed in parallel. Furthermore, some of these protocol use a constant number of rounds.
- Some zero-knowledge proofs (for NP) preserve their security when many copies are executed concurrently, but such a result is not known for constant-round protocols.

The second basic question regarding zero-knowledge refers to the usage of the adversary's program within the proof of security (i.e., demonstration of the zero-knowledge property). For 15 years, all known proofs of security used the adversary's program as a black-box (i.e., a universal simulator was presented using the adversary's program as an oracle). Furthermore, it was believed that there is no advantage in having access to the code of the adversary's program. Consequently it was conjectured that negative results regarding black-box simulation represent an inherent limitation of zero-knowledge. This believe has been refuted recently by a zero-knowledge argument (for NP) that has important properties that are unachievable by black-box simulation. For example, this zero-knowledge argument uses a constant number of rounds and preserves its security when an (a-priori fixed polynomial) number of copies are executed concurrently.[1]

The composability of zero-knowledge proofs is discussed in Section 5 and the use of the adversary's program within the proof of security is discussed in Section 6. Other topics treated in the second part of this tutorial include proofs of knowledge (Section 7), Non-Interactive Zero-Knowledge proofs (Section 8), Statistical Zero-Knowledge (Section 9), Knowledge Complexity (Section 10), and resettability of a party's random-tape (Section 11).

### 1.3 Comments

The notion of zero-knowledge has had a vast impact on the development of cryptography. In particular, zero-knowledge proofs of various types were explicitly used (as a tool) in a variety of applications. We wish to highlight also the indirect impact of zero-knowledge on the definitional approach underlying the foundations of cryptography (cf. Section 3.1). In addition, zero-knowledge has served as a source of inspiration for complexity theory (cf. Section 13).

**A Brief Historical Account** (regarding the main part of the tutorial)**:** The concept of zero-knowledge has been introduced by Goldwasser, Micali, and Rackoff [66]. Although their work, which also introduced interactive proof systems, has first appeared in *STOC95*, early versions of it have existed as early as 1982 (and were rejected three times from major conferences; i.e., *FOCS83*, *STOC84*, and *FOCS84*). The wide applicability of zero-knowledge proofs was first demonstrated by Goldreich, Micali and Wigderson, who showed how to construct zero-knowledge proof systems for any NP-set, using any commitment scheme [57]. An important technique for the design of zero-knowledge was introduced by Feige, Lapidot and Shamir [38], based on the notion of witness indistinguishability (which was introduced by Feige and Shamir [39]). Important contributions to the study of the sequential, parallel and concurrent composition of zero-knowledge protocols were presented in [55, 59], [55, 51] and [34, 81, 28, 72, 7], respectively. The power of non-black-box simulators has been recently discovered by Barak [7].

[1]This result falls short of achieving a fully concurrent zero-knowledge argument, because the number of concurrent copies must be fixed before the protocol is presented. Specifically, the protocol uses messages that are longer than the allowed number of concurrent copies.

**Suggestions for further reading:** For further details regarding most of the material, the reader is referred to [49, Chap. 4]. For a wider perspective on probabilistic proof systems, the reader is referred to [48, Chap. 2].

# Part I
# The Basics

## 2 Preliminaries

Modern Cryptography, is concerned with the construction of *efficient* schemes for which it is *infeasible* to violate the security feature. The same concern underlies the main definitions of zero-knowledge. Thus, for starters, we need a notion of efficient computations as well as a notion of infeasible ones. The computations of the legitimate users of the scheme ought be efficient, whereas violating the security features (via an adversary) ought to be infeasible.

Efficient computations are commonly modeled by computations that are polynomial-time in the security parameter. The polynomial bounding the running-time of the legitimate user's strategy is fixed and typically explicit (and small). Here (i.e., when referring to the complexity of the legitimate users) we are in the same situation as in any algorithmic setting. Things are different when referring to our assumptions regarding the computational resources of the adversary. A common approach is to postulate that the latter are polynomial-time too, where the polynomial is *not a-priori specified*. In other words, the adversary is restricted to the class of efficient computations and anything beyond this is considered to be infeasible. Although many definitions explicitly refer to this convention, this convention is *inessential* to any of the results known in the area. In all cases, a more general statement can be made by referring to adversaries of running-time bounded by any super-polynomial function (or class of functions). Still, for sake of concreteness and clarity, we shall use the former convention in our treatment.

Actually, in order to simplify our exposition, we will often consider as infeasible any computation that cannot be conducted by a (possibly non-uniform) family of polynomial-size circuits. For simplicity we consider families of circuits $\{C_n\}$, where for some polynomials $p$ and $q$, each $C_n$ has exactly $p(n)$ input bits and has size at most $q(n)$.

*Randomized computations* play a central role in the definition of zero-knowledge (as well as in cryptography at large). That is, we allow the legitimate users to employ randomized computations, and likewise we consider adversaries that employ randomized computations. This brings up the issue of success probability: typically, we require that legitimate users succeed (in fulfilling their legitimate goals) with probability 1 (or negligibly close to this), whereas adversaries succeed (in violating the security features) with negligible probability. Thus, the notion of a negligible probability plays an important role in our exposition. One feature required of the definition of *negligible probability* is to yield a robust notion of rareness: A rare event should occur rarely even if we repeat the experiment for a feasible number of times. Likewise, we consider two events to occur "as frequently" if the absolute difference between their corresponding occurrence probabilities is negligible.

### 2.1 Interactive proofs and argument systems

Before defining zero-knowledge proofs, we have to define proofs. The standard notion of static (i.e., non-interactive) proofs will not do, because static zero-knowledge proofs exist only for sets that are easy to decide (i.e, are in $\mathcal{BPP}$) [59] whereas we are interested in zero-knowledge proofs for arbitrary NP-sets. Instead, we use the notion of an interactive proof (introduced exactly for that reason by Goldwasser, Micali and Rackoff [66]). That is, here a proof is a (multi-round) randomized protocol for two parties, called *verifier* and *prover*, in which the prover wishes to convince the verifier of the validity of a given assertion. Such an *interactive proof* should allow the prover to convince the verifier of the validity of any true assertion, whereas NO prover strategy may fool the verifier to accept false assertions. Both the above *completeness* and *soundness* conditions should hold with high probability (i.e., a negligible error probability is allowed).

We comment that interactive proofs emerge naturally when associating the notion of efficient verification, which underlies the notion of a proof system, with probabilistic and interactive polynomial-time computations. This association is quite natural in light of the growing acceptability of randomized and distributed computations. Thus, a "proof" in this context is not a fixed and static object, but rather a randomized and dynamic (i.e., interactive) process in which the verifier interacts with the prover. Intuitively, one may think of this interaction as consisting of "tricky" questions asked by the verifier, to which the prover has to reply "convincingly". The above discussion, as well as the following definition, makes explicit reference to a prover, whereas a prover is only implicit in the traditional definitions of proof systems (e.g., NP-proofs).

Loosely speaking, an interactive proof is a game between a computationally bounded verifier and a computationally unbounded prover whose goal is to convince the verifier of the validity of some assertion. Specifically, the verifier is probabilistic polynomial-time. It is required that if the assertion holds then the verifier always accepts (i.e., when interacting with an appropriate prover strategy). On the other hand, if the assertion is false then the verifier must reject with "noticeable" probability, no matter what strategy is being employed by the prover. Indeed, the error probability (in the soundness condition) can be reduced by (either sequential or parallel) repetitions.

**Definition 1** (Interactive Proof systems and the class $\mathcal{IP}$ [66]): *An* interactive proof system for a set *$S$ is a two-party game, between a* verifier *executing a probabilistic polynomial-time strategy* (denoted $V$) *and a* prover *which executes a computationally unbounded strategy* (denoted $P$)*, satisfying*

- Completeness*: For every $x \in S$ the verifier $V$ always accepts after interacting with the prover $P$ on common input $x$.*
- Soundness*: For some polynomial $p$, it holds that for every $x \notin S$ and every potential strategy $P^*$, the verifier $V$ rejects with probability at least $1/p(|x|)$, after interacting with $P^*$ on common input $x$.*

*The class of problems having interactive proof systems is denoted $\mathcal{IP}$.*

Note that by repeating such a proof system for $O(p(|x|)^2)$ times, we may decrease the probability that $V$ accepts a false statement (from $1-(1/p(|x|))$) to $2^{-p(|x|)}$. Thus, when constructing interactive proofs we sometimes focus on obtaining a noticeable rejection probability for NO-instances (i.e., obtaining soundness error bounded away from 1), whereas when using interactive proofs we typically assume that their soundness error is negligible.

**Variants:** Arthur-Merlin games (a.k.a *public-coin* proof systems), introduced by Babai [4], are a special case of interactive proofs, where the verifier must send the outcome of any coin it tosses (and thus need not send any other information). Yet, as shown in [67], this restricted case has essentially the same power as the general case (introduced by Goldwasser, Micali and Rackoff [66]). Thus, in the context of interactive proof systems, asking random questions is as powerful as asking "tricky" ones. (As we shall see, this does not necessarily hold in the context of zero-knowledge proofs.) Also, in some sources interactive proofs are defined so that two-sided error probability is allowed (rather than requiring "perfect completeness" as done above); yet, this does not increase their power [44].

**Arguments (or Computational Soundness):** A fundamental variant on the notion of interactive proofs was introduced by Brassard, Chaum and Crépeau [21], who relaxed the soundness condition so that it only refers to feasible ways of trying to fool the verifier (rather than to all possible ways). Specifically, the soundness condition was replaced by the following computational soundness condition that asserts that it is infeasible to fool the verifier into accepting false statements.

> For every polynomial $p$, every prover strategy that is implementable by a family of polynomial-size circuits $\{C_n\}$, and every sufficiently large $x \in \{0,1\}^* \setminus S$, the probability that $V$ accepts $x$ when interacting with $C_{|x|}$ is less than $1/p(|x|)$.

We warn that although the computational-soundness error can always be reduced by sequential repetitions, it is not true that this error can always be reduced by parallel repetitions (cf. [14]). Protocols that satisfy the computational-soundness condition are called arguments.[2] We mention that argument systems may be more efficient than interactive proofs (see [70, 53]).

**Terminology.** Whenever we wish to blur the distinction between proofs and arguments, we will use the term protocols. We will consider such a protocol trivial if it establishes membership in a BPP-set (because membership in such a set can be determined by the verifier itself). On the other hand, we will sometimes talk about protocols for a $\mathcal{NP}$, when what we mean is protocols for each set in $\mathcal{NP}$. (This terminology is quite common in the area.)

## 2.2 Computational Difficulty and One-Way Functions

Most positive results regarding zero-knowledge proofs are based on intractability assumptions. Furthermore, the very notion of a zero-knowledge proof is interesting only in case the assertion being proven valid is hard to verify in probabilistic polynomial-time. Thus, our discussion always assumes at least implicitly that $\mathcal{IP}$ is not contained in $\mathcal{BPP}$, and often we explicitly assume more than that.

In general, Modern Cryptography is concerned with the construction of schemes that are easy to operate (properly) but hard to foil. Thus, a complexity gap (i.e., between the complexity of proper usage and the complexity of defeating the prescribed functionality) lies in the heart of Modern Cryptography. However, gaps as required for Modern Cryptography are not known to exist; they are only widely believed to exist. Indeed, almost all of Modern Cryptography rises or falls with the question of whether one-way functions exist. One-way functions are functions that are easy to evaluate but hard (on the average) to invert (cf. [32]). That is, a function $f:\{0,1\}^* \to \{0,1\}^*$ is called one-way if there is an efficient algorithm that on input $x$ outputs $f(x)$, whereas any feasible algorithm that tries to find a preimage of $f(x)$ under $f$ may succeed only with negligible probability (where the probability is taken uniformly over the choices of $x$ and the algorithm's coin tosses). Associating feasible computations with (possibly non-uniform) families of polynomial-size circuits, we obtain the following definition.

[2] A related notion not discussed here is that of CS-proofs, introduced by Micali [75].

**Definition 2** (one-way functions): *A function* $f:\{0,1\}^* \to \{0,1\}^*$ *is called* one-way *if the following two conditions hold:*

1. easy to evaluate: *There exist a polynomial-time algorithm* $A$ *such that* $A(x) = f(x)$ *for every* $x \in \{0,1\}^*$.
2. hard to invert: *For every family of polynomial-size circuits* $\{C_n\}$*, every polynomial* $p$*, and all sufficiently large* $n$*,*

$$\Pr[C_n(f(x)) \in f^{-1}(f(x))] < \frac{1}{p(n)}$$

*where the probability is taken uniformly over all the possible choices of* $x \in \{0,1\}^n$.

Some of the most popular candidates for one-way functions are based on the conjectured intractability of computational problems in number theory. One such conjecture is that it is infeasible to factor large integers. Consequently, the function that takes as input two (equal length) primes and outputs their product is widely believed to be a one-way function.

**Terminology.** Some of the (positive) results mentioned below require stronger forms of one-way functions (e.g., one-way permutations with (or without) trapdoor [49, Sec. 2.4.4] and claw-free permutation pairs [49, Sec. 2.4.5]). Whenever we wish to avoid the specific details, we will talk about standard intractability assumptions. In all cases, the conjectured intractability of factoring will suffice.

### 2.3 Computational Indistinguishability

A central notion in Modern Cryptography is that of "effective similarity" (introduced by Goldwasser, Micali and Yao [65, 85]). The underlying idea is that we do not care whether or not objects are equal, all we care is whether or not a difference between the objects can be observed by a feasible computation. In case the answer is negative, the two objects are equivalent as far as any practical application is concerned. Indeed, like in many other cryptographic definitions, in the definition of general/computational zero-knowledge we will freely interchange such (computationally indistinguishable) objects.

The asymptotic formulation of computational indistinguishability refers to (pairs of) probability ensembles, which are infinite sequences of finite distributions, rather than to (pairs of) finite distributions. Specifically, we consider sequences indexed by strings (rather than by integers (in unary representation)). For $S \subseteq \{0,1\}^*$, we consider the probability ensembles $X = \{X_\alpha\}_{\alpha\in S}$ and $Y = \{Y_\alpha\}_{\alpha\in S}$, where each $X_\alpha$ (resp., $Y_\alpha$) is a distribution that ranges over strings of length polynomial in $|\alpha|$. We say that $X$ and $Y$ are computationally indistinguishable if for every feasible algorithm $A$ the difference $d_A(n) \stackrel{\rm def}{=} \max_{\alpha\in\{0,1\}^n}\{|\Pr[A(X_\alpha)=1] - \Pr[A(Y_\alpha)=1]|\}$ is a negligible function in $|\alpha|$. That is:

**Definition 3** (computational indistinguishability [65, 85]): *We say that* $X = \{X_\alpha\}_{\alpha\in S}$ *and* $Y = \{Y_\alpha\}_{\alpha\in S}$ *are* computationally indistinguishable *if for every family of polynomial-size circuits* $\{D_n\}$*, every polynomial* $p$*, all sufficiently large* $n$ *and every* $\alpha \in \{0,1\}^{\mathrm{poly}(n)} \cap S$*,*

$$|\Pr[D_n(X_\alpha)=1] - \Pr[D_n(Y_\alpha)=1]| < \frac{1}{p(n)}$$

*where the probabilities are taken over the relevant distribution* (i.e., either $X_n$ or $Y_n$).

That is, think of $D = \{D_n\}$ as of somebody who wishes to distinguish two distributions (based on a sample given to it), and think of 1 as of $D$'s verdict that the sample was drawn according to the first distribution. Saying that the two distributions are computationally indistinguishable means that if $D$ is an efficient procedure then its verdict is not really meaningful (because the verdict is almost as often 1 when the input is drawn from the first distribution as when the input is drawn from the second distribution).

We comment that indistinguishability by a single sample (as defined above) implies indistinguishability by multiple samples. Also note that the definition would not have been stronger if we were to provide the distinguisher (i.e., $D$) with the index (i.e., $\alpha$) of the distribution-pair being tested.[3]

# 3 Definitional Issues

Loosely speaking, zero-knowledge proofs are proofs that yield nothing beyond the validity of the assertion. That is, a verifier obtaining such a proof only gains conviction in the validity of the assertion. This is formulated by saying that anything that can be feasibly obtained from a zero-knowledge proof is also feasibly computable from the (valid) assertion itself. The latter formulation follows the simulation paradigm, which is discussed next.

## 3.1 The Simulation Paradigm

In defining zero-knowledge proofs, we view the verifier as a potential adversary that tries to gain knowledge from the (prescribed) prover. We wish to state that no (feasible) adversary strategy for the verifier can gain anything from the prover (beyond conviction in the validity of the assertion). Let us consider the desired formulation from a wide perspective.

A key question regarding the modeling of security concerns is how to express the intuitive requirement that an adversary "gains nothing substantial" by deviating from the prescribed behavior of an honest user. Our approach is that the adversary *gains nothing* if whatever it can obtain by unrestricted adversarial behavior can be obtained within essentially the same computational effort by a benign behavior. The definition of the "benign behavior" captures what we want to achieve in terms of security, and is specific to the security concern to be addressed. For example, in the previous paragraph, we said that a proof is zero-knowledge if it yields nothing beyond the validity of the assertion (i.e., the benign behavior is any computation that is based (only) on the assertion itself, while assuming that the latter is valid). Thus, in a zero-knowledge proof no feasible adversarial strategy for the verifier can obtain more than a "benign verifier", which believes the assertion, can obtain from the assertion itself. We comment that the simulation paradigm, which was first developed in the context of zero-knowledge [66], is pivotal also to the definition of the security of encryption schemes (cf. [50, Chap. 5]) and cryptographic protocols (cf. [24, 47]).

A notable property of defining security (or zero-knowledge) via the simulation paradigm is that this approach is "overly liberal" with respect to its view of the abilities of the adversary as well as to what might constitute a gain for the adversary. Thus, the approach may be considered overly cautious, because it prohibits also "non-harmful" gains of some "far fetched" adversaries. We warn against this impression. Firstly, there is nothing more dangerous in cryptography than to consider "reasonable" adversaries (a notion which is almost a contradiction in terms): typically, the

[3] Furthermore, the definition would not have been stronger if we were to consider a specialized polynomial-size circuit per each $\alpha \in S$ (i.e., consider the difference $|\mathsf{Pr}[D_\alpha(X_\alpha)=1] - \mathsf{Pr}[D_\alpha(Y_\alpha)=1]|$ for any set of circuits $D = \{D_\alpha\}_{\alpha \in S}$ such that the size of $D_\alpha$ is polynomial in $|\alpha|$).

adversaries will try exactly what the system designer has discarded as "far fetched". Secondly, it seems impossible to come up with definitions of security that distinguish "breaking the scheme in a harmful way" from "breaking it in a non-harmful way": what is harmful is application-dependent, whereas a good definition of security ought to be application-independent (as otherwise using the scheme in any new application will require a full re-evaluation of its security). Furthermore, even with respect to a specific application, it is typically very hard to classify the set of "harmful breakings".

### 3.2 The Basic Definition

Zero-knowledge is a property of some prover strategies. More generally, zero-knowledge is a property of some interactive machines. Fixing an interactive machines (e.g., prescribed provers), we consider what can be computed by an *arbitrary* feasible adversary (e.g., verifier) that interacts with the fixed machine on a common input taken from a predetermined set (in our case the set of valid assertions). This is compared against what can be computed by an *arbitrary* feasible algorithm that is only given the input itself. A strategy $A$ is zero-knowledge on (inputs from) the set $S$ if, for every feasible strategy $B^*$, there exists a feasible computation $C^*$ so that the following two probability ensembles are computationally indistinguishable:

1. $\{(A,B^*)(x)\}_{x\in S} \stackrel{\rm def}{=}$ the output of $B^*$ after interacting with $A$ on common input $x \in S$; and
2. $\{C^*(x)\}_{x\in S} \stackrel{\rm def}{=}$ the output of $C^*$ on input $x \in S$.

We stress that the first ensemble represents an actual execution of an interactive protocol, whereas the second ensemble represents the computation of a stand-alone procedure (called the "simulator"), which does not interact with anybody. Thus, whatever can be feasibly extracted from interaction with $A$ on input $x \in S$, can also be feasibly extracted from $x$ itself. This means that nothing was gain by the interaction itself (beyond confidence in the assertion $x \in S$).

The above definition does NOT account for auxiliary information that an adversary may have prior to entering the interaction. Accounting for such auxiliary information is essential for using zero-knowledge proofs as subprotocols inside larger protocols (see [55, 59]). This is taken care of by a more strict notion called auxiliary-input zero-knowledge.[4]

**Definition 4** (zero-knowledge [66], revisited [59]): *A strategy $A$ is* auxiliary-input zero-knowledge *on inputs from $S$ if for every probabilistic polynomial-time strategy $B^*$ and every polynomial $p$ there exists a probabilistic polynomial-time algorithm $C^*$ such that the following two probability ensembles are computationally indistinguishable:*

1. *$\{(A,B^*(z))(x)\}_{x\in S\,,\,z\in\{0,1\}^{p(|x|)}} \stackrel{\rm def}{=}$ the output of $B^*$ when having auxiliary-input $z$ and interacting with $A$ on common input $x \in S$; and*
2. *$\{C^*(x,z)\}_{x\in S\,,\,z\in\{0,1\}^{p(|x|)}} \stackrel{\rm def}{=}$ the output of $C^*$ on inputs $x \in S$ and $z \in \{0,1\}^{p(|x|)}$.*

[4] We note that the following definition seems stronger than merely allowing the verifier and simulator to be arbitrary polynomial-size circuits. The issue is that the latter formulation does not guarantee that the simulator can be easily derived from the cheating verifier nor that the length of the simulator's description is related to the length of the description of the verifier. Both issues are important when trying to use zero-knowledge proofs as subprotocols inside larger protocols or to compose them (even sequentially). For further discussion, see Section 5.

*An interactive proof* (resp., an argument) *system for $S$ is called* auxiliary-input zero-knowledge *if the prescribed prover strategy is auxiliary-input zero-knowledge on inputs from $S$.*[5]

The more basic definition of zero-knowledge is obtained by eliminating the auxiliary-input $z$ from Definition 4. We comment that almost all known zero-knowledge proofs are in fact auxiliary-input zero-knowledge. (Notable exceptions are zero-knowledge proofs constructed on purpose in order to show a separation between these two notions; e.g., in [55].)

We stress that the zero-knowledge property of an interactive proof (resp., argument) refers to all feasible *adversarial strategies that the verifier may employ* (in attempt to extract knowledge from the prescribed prover that tries to convince the verifier to accept a valid assertion). In contrast, the soundness property of an interactive proof (resp., the computational-soundness property of an argument) refers to all possible (resp., feasible) *adversarial strategies that the prover may employ* (in attempt to fool the prescribed verifier to accept a false assertion). Finally, the completeness property refers to the behavior of both prescribed strategies (when given, as common input, a valid assertion).

## 3.3 Variants

The reader may skip the current subsection and return to it whenever encountering (especially in the second part of this tutorial) a notion that was not defined above.

### 3.3.1 Universal and black-box simulation

We have already discussed two variants on the basic definition (i.e., with or without auxiliary-inputs). Further strengthening of Definition 4 is obtained by requiring the existence of a universal simulator, denoted $\mathcal{C}$, that is given the program of the verifier (i.e., $B^*$) as an auxiliary-input; that is, in terms of Definition 4, one should replace $C^*(x, z)$ by $\mathcal{C}(x, z, \langle B^* \rangle)$, where $\langle B^* \rangle$ denotes the description of the program of $B^*$ (which may depend on $x$ and on $z$).[6] That is, we effectively restrict the simulation by requiring that it be a uniform (feasible) function of the verifier's program (rather than arbitrarily depend on it). This restriction is very natural, because it seems hard to envision an alternative way of establishing the zero-knowledge property of a given protocol.

Taking another step, one may argue that since it seems infeasible to reverse-engineer programs, the simulator may as well just use the verifier strategy as an oracle (or as a "black-box"). This reasoning gave rise to the notion of black-box simulation, which was introduced and advocated in [55] and further studied in numerous works (see, e.g., [28]). The belief was that impossibility results regarding black-box simulation represent inherent limitations of zero-knowledge itself. However, this belief has been refuted recently by Barak [7]. For further discussion, see Section 6.

### 3.3.2 Honest verifier versus general cheating verifier

The (general) definition of zero-knowledge (i.e., Definition 4) refers to all feasible verifier strategies. This choice is most natural since zero-knowledge is supposed to capture the robustness of the prover under *any feasible* (i.e., adversarial) attempt to gain something by interacting with it. Thus, we typically view the verifier as an adversary that is trying to cheat.

[5]Note that the *prescribed* verifier strategy (which is a probabilistic polynomial-time strategy that only depends on the common input) is always auxiliary-input zero-knowledge. In contrast, typical prover strategies are implemented by probabilistic polynomial-time algorithms that are given an auxiliary input (which is not given to the verifier), but not by probabilistic polynomial-time algorithms that are only given the common input.

[6]Actually, we may incorporate $x$ and $z$ in $\langle B^* \rangle$, and thus replace $\mathcal{C}(x, z, \langle B^* \rangle)$ by $\mathcal{C}(\langle B^* \rangle)$.

A weaker and still interesting notion of zero-knowledge refers to what can be gained by an "honest verifier" (or rather a semi-honest verifier)[7] that interacts with the prover as directed, with the exception that it may maintain (and output) a record of the entire interaction (i.e., even if directed to erase all records of the interaction). Although such a weaker notion is not satisfactory for standard cryptographic applications, it yields a fascinating notion from a conceptual as well as a complexity-theoretic point of view. Furthermore, as shown in [62], every public-coin proof system that is *zero-knowledge with respect to the honest-verifier* can be transformed into a *standard zero-knowledge* proof that maintains many of the properties of the original protocol (and without increasing the prover's powers or using any intractability assumptions).

We stress that the definition of zero-knowledge with respect to the honest-verifier $V$ is derived from Definition 4 by considering a single verifier strategy $B$ that is equal to $V$ except that $B$ also maintains a record of the entire interaction (including its own coin tosses) and outputs this record at the end of the interaction. (Thus, all messages sent by $B$ are equal to the messages that would have been sent by $V$.)

### 3.3.3 Statistical versus Computational Zero-Knowledge

Recall that the definition of zero-knowledge postulates that for every probability ensemble of one type (i.e., representing the verifier's output after interaction with the prover) there exists a "similar" ensemble of a second type (i.e., representing the simulator's output). One key parameter is the interpretation of "similarity". Three interpretations, yielding different notions of zero-knowledge, have been commonly considered in the literature (cf., [66, 42]):

1. Perfect Zero-Knowledge (PZK) requires that the two probability ensembles be identical.[8]

2. Statistical Zero-Knowledge (SZK) requires that these probability ensembles be statistically close (i.e., the variation distance between them is negligible).

3. Computational (or rather general) Zero-Knowledge (CZK) requires that these probability ensembles be computationally indistinguishable.

Indeed, Computational Zero-Knowledge (CZK) is the most liberal notion, and is the notion considered in Definition 4 as well as in most of this tutorial. (In particular, whenever we fail to qualify the type of zero-knowledge, we mean computational zero-knowledge.) The only exception is Section 9, which is devoted to a discussion of Statistical (or Almost-Perfect) Zero-Knowledge (SZK). We note that the class SZK contains several problems that are considered intractable.

### 3.3.4 Strict versus expected probabilistic polynomial-time

So far, we did not specify what we exactly mean by the term probabilistic polynomial-time. Two common interpretations are:

1. Strict probabilistic polynomial-time. That is, there exist a (polynomial in the length of the input) bound on the *number of steps in each possible run* of the machine, regardless of the outcome of its coin tosses.

[7]The term "honest verifier" is more appealing when considering an alternative (equivalent) formulation of Definition 4. In the alternative definition, the simulator is "only" required to generate the verifier's view of the real interaction, when the verifier's view includes its inputs, the outcome of its coin tosses, and all messages it has received.

[8]The actual definition of PZK allows the simulator to fail (while outputting a special symbol) with some probability that is bounded away from 1, and the output distribution of the simulator is conditioned on its not failing.

2. Expected probabilistic polynomial-time. The standard approach is to look at the running-time as a random variable and *bound its expectation* (by a polynomial in the length of the input). As observed by Levin [73] (cf. [46]), this definitional approach is quite problematic (e.g., it is not model-independent and is not closed under algorithmic composition), and an alternative treatment of this random variable is preferable.[9]

Consequently, the notion of expected polynomial-time raises a variety of conceptual and technical problems. For that reason, whenever possible, one should prefer to use the more robust (and restricted) notion of strict (probabilistic) polynomial-time. Thus, with the *exception of constant-round* zero-knowledge protocols, whenever we talk of a probabilistic polynomial-time verifier (resp., simulator) we mean one in the strict sense. In contrast, with the exception of [7, 11] and [40],[10] all results regarding *constant-round* zero-knowledge protocols refer to a strict polynomial-time verifier and an expected polynomial-time simulator, which is indeed a small cheat. For further discussion, the reader is referred to [11].

# 4 Zero-Knowledge Proofs for every NP-set

A question avoided so far is whether zero-knowledge proofs exist at all. Clearly, every set in $\mathcal{P}$ (or rather in $\mathcal{BPP}$)[11] has a "trivial" zero-knowledge proof (in which the verifier determines membership by itself); however, what we seek is zero-knowledge proofs for statements that the verifier cannot decide by itself.

## 4.1 Constructing Zero-Knowledge Proofs for NP-sets

Assuming the existence of commitment schemes[12], which in turn exist if one-way functions exist [76, 68], *there exist* (auxiliary-input) *zero-knowledge proofs of membership in any NP-set* (i.e., sets having efficiently verifiable static proofs of membership). These zero-knowledge proofs, first constructed by Goldreich, Micali and Wigderson [57] (and depicted in Figure 1), have the following important property: the prescribed prover strategy is efficient, provided it is given as auxiliary-input an NP-witness to the assertion (to be proven). That is:

**Theorem 5** ([57], using [68, 76]): *If one-way functions exist then every set $S \in \mathcal{NP}$ has a zero-knowledge interactive proof. Furthermore, the prescribed prover strategy can be implemented in probabilistic polynomial-time, provided it is given as auxiliary-input an NP-witness for membership of the common input in $S$.*

[9] Specifically, it is preferable to define expected polynomial-time as having running time that is polynomially-related to a function that has linear expectation. That is, rather than requiring that $\mathsf{E}[X_n] = \mathrm{poly}(n)$, one requires that for some $Y_n$ it holds that $X_n = \mathrm{poly}(Y_n)$ and $\mathsf{E}[Y_n] = O(n)$. The advantage of the latter approach is that if $X_n$ is deemed *polynomial on the average* then so is $X_n^2$, which is not the case under the former approach (e.g., $X_n = 2^n$ with probability $2^{-n}$ and $X_n = n$ otherwise).

[10] Specifically, in [7, 11] both the verifier and the simulator run in strict polynomial-time, whereas in [40] both run in expected polynomial-time. We comment that, as shown in [11], the use of non-black-box is necessary for the non-triviliaty of constant-round zero-knowledge protocols under the strict definition.

[11] Trivial zero-knowledge proofs for sets in $\mathcal{BPP} \setminus \mathrm{co}\mathcal{RP}$ require modifying the definition of interactive proofs such that to allow a negligible error also in the completeness condition. Alternatively, zero-knowledge proofs for sets in $\mathcal{BPP}$ can be constructed by having the prover send a single message that is distributed almost uniformly (cf. [44]).

[12] Loosely speaking, commitment schemes are digital analogue of non-transparent sealed envelopes. See further discussion in Figure 1.

Theorem 5 makes zero-knowledge a very powerful tool in the design of cryptographic schemes and protocols (see below). We comment that the intractability assumption used in Theorem 5 seems essential; see [78].

Commitment schemes are digital analogies of sealed envelopes (or, better, locked boxes). Sending a commitment means sending a string that binds the sender to a unique value without revealing this value to the receiver (as when getting a locked box). Decommitting to the value means sending some auxiliary information that allows to read the unique committed value (as when sending the key to the lock).

**Common Input:** A graph $G(V, E)$. Suppose that $V \equiv \{1, ..., n\}$ for $n \stackrel{\text{def}}{=} |V|$.

**Auxiliary Input (to the prover):** A 3-coloring $\phi : V \rightarrow \{1, 2, 3\}$.

The following 4 steps are repeated $t \cdot |E|$ many times so to obtain $\exp(-t)$ soundness error.

**Prover's first step (P1):** Select uniformly a permutation $\pi$ over $\{1, 2, 3\}$. For $i = 1$ to $n$, send the verifier a commitment to $\pi(\phi(i))$.

**Verifier's first step (V1):** Select uniformly an edge $e \in E$ and send it to the prover.

**Prover's second step (P2):** Upon receiving $e = (i, j) \in E$, decommit to the $i$th and $j$th values sent in Step (P1).

**Verifier's second step (V2):** Check whether or not the decommitted values are different elements of $\{1, 2, 3\}$ and whether or not they match the commitments received in Step (P1).

Figure 1: The zero-knowledge proof of Graph 3-Colorability (of [57]). Zero-knowledge proofs for other NP-sets can be obtained using the standard reductions.

**About the protocol of Figure 1.** Let us consider a single execution of the main loop. Clearly, the prescribed prover is implemented in probabilistic polynomial-time, and always convinces the verifier (provided that it is given a valid 3-coloring of the common input graph). In case the graph is not 3-colorable then, no matter how the prover behaves, the verifier will reject with probability at least $1/|E|$ (because at least one of the edges must be improperly colored by the prover). We stress that the verifier selects uniformly which edge to inspect after the prover has committed to the colors of all vertices. Thus, Figure 1 depicts an interactive proof system for Graph 3-Colorability. As can be expected, the zero-knowledge property is the hardest to establish, and we will confine ourselves to presenting a simulator (which we hope will convince the reader without a detailed analysis). We start with three simplifying conventions (which are useful in general):

1. Without loss of generality, we may assume that the cheating verifier strategy is implemented by a *deterministic* polynomial-size circuit (or, equivalently, by a polynomial-time algorithm with an auxiliary input). This is justified by fixing any outcome of the verifier's coins, and observing that our (uniform) simulation of the various residual deterministic strategies yields a simulation of the original probabilistic strategy.

2. Without loss of generality, it suffices to consider cheating verifiers that output the full transcript of the interaction. This is justified by observing that the output of the original verifier can be computed by an algorithm of comparable complexity that is given the full transcript of the interaction.

3. Without loss of generality, it suffices to construct a simulator that produces output with some noticeable probability. This is the case because, by repeatedly invoking this simulator (polynomially) many times, we may obtain a simulator that fails to produce an output with negligible probability, and the latter yields a simulator as required.

The simulator starts by selecting uniformly and independently a random color (i.e., element of $\{1, 2, 3\}$) for each vertex, and feeding the verifier strategy with random commitments to these random colors. Indeed, the simulator feeds the verifier with a distribution that is very different from the distribution the verifier sees in an interaction with the prover. However, being computationally-restricted the verifier cannot tell these distributions apart (or else we obtain a contradiction to the security of the commitment scheme in use). Now, if the verifier asks to inspect an edge that is properly colored then the simulator performs the proper decommitment action and outputs the transcript of this interaction. Otherwise, the simulator halts proclaiming failure. We claim that failure occurs with probability approximately $1/3$ (or else we obtain a contradiction to the security of the commitment scheme in use). Furthermore, based on the same hypothesis (but via a more complex proof), conditioned on not failing, the output of the simulator is computationally indistinguishable from the transcript of the real interaction.

**Zero-knowledge proofs for other NP-sets.** By using the standard Karp-reductions to 3-Colorability, the protocol of Figure 1 can be used for constructing zero-knowledge proofs for any set in $\mathcal{NP}$. We comment that this is probably the first time that an NP-completeness result was used in a "positive" way (i.e., in order to construct something rather than in order to derive a hardness result). Subsequent positive uses of completeness results have appeared in the context of interactive proofs [74, 84], probabilistically checkable proofs [5, 36, 3, 2], "hardness versus randomness trade-offs" [6], and statistical zero-knowledge [83].

**Efficiency considerations.** The protocol in Figure 1 calls for invoking some constant-round protocol for a non-constant number of times. At first sight, it seems that one can derive a constant-round zero-knowledge proof system (of negligible soundness error) by performing these invocations in parallel (rather than sequentially). Unfortunately, as demonstrated in [55], this intuition is not sound. See further discussions in Sections 5 and 6. We note that the number of rounds in a protocol is commonly considered the most important efficiency criteria (or complexity measure), and typically one desires to have it be a constant. We mention that, under standard intractability assumptions (e.g., the intractability of factoring), constant-round zero-knowledge proofs (of negligible soundness error) exists for every set in $\mathcal{NP}$ (cf. [54]).

## 4.2 Using Zero-Knowledge Proofs for NP-sets

We stress two important aspects regarding Theorem 5: Firstly, it provides a zero-knowledge proof for every NP-set, and secondly the prescribed prover can be implemented in probabilistic polynomial-time when given an adequate NP-witness.

**A generic application.** In a typical cryptographic setting, a user referred to as $U$, has a secret and is supposed to take some action depending on its secret. The question is how can other users verify that $U$ indeed took the correct action (as determined by $U$'s secret and the publicly known information). Indeed, if $U$ discloses its secret then anybody can verify that $U$ took the correct action. However, $U$ does not want to reveal its secret. Using zero-knowledge proofs we can satisfy both conflicting requirements (i.e., having other users verify that $U$ took the correct action without violating $U$'s interest in not revealing its secrets). That is, $U$ can prove in zero-knowledge that it took the correct action. Note that $U$'s claim to having taken the correct action is an NP-assertion (since $U$'s legal action is determined as a polynomial-time function of its secret and the public information), and that $U$ has an NP-witness to its validity (i.e., the secret is an NP-witness to the claim that the action fits the public information). Thus, by Theorem 5, it is possible for $U$ to efficiently prove the correctness of its action without yielding anything about its secret. Consequently, it is fair to ask $U$ to prove (in zero-knowledge) that it behaves properly, and so force $U$ to behave properly. Indeed, "forcing proper behavior" is the canonical application of zero-knowledge proofs (see [58, 47]).

**Zero-knowledge proofs for all IP.** For the sake of elegancy, we mention that under the same assumption used in case of $\mathcal{NP}$, it holds that *any set that has an interactive proof also has a zero-knowledge interactive proof* (cf. [69, 15]).

# Part II
# Advanced Topics

## 5 Composing zero-knowledge protocols

A natural question regarding zero-knowledge proofs (and arguments) is whether the zero-knowledge condition is preserved under a variety of composition operations. Three types of composition operation were considered in the literature: *sequential composition*, *parallel composition* and *concurrent composition*. We note that the preservation of zero-knowledge under these forms of composition is not only interesting on its own sake, but rather also sheds light of the preservation of the security of general protocols under these forms of composition.

We stress that when we talk of composition of protocols (or proof systems) we mean that the honest users are supposed to follow the prescribed program (specified in the protocol description) that refers to a single execution. That is, the actions of honest parties in each execution are independent of the messages they received in other executions. The adversary, however, may coordinate the actions it takes in the various executions, and in particular its actions in one execution may depend also on messages it received in other executions.

Let us motivate the asymmetry between the independence of executions assumed of honest parties but not of the adversary. Coordinating actions in different executions is typically difficult but not impossible. Thus, it is desirable to use composition (as defined above) rather than to use protocols that include inter-execution coordination-actions, which require users to keep track of all executions that they perform. Actually, trying to coordinate honest executions is even more problematic than it seems because one may need to coordinate executions of *different* honest parties (e.g., all employees of a big cooperation or an agency under attack), which in many cases is highly unrealistic. On the other hand, the adversary attacking the system may be willing to go into the extra trouble of coordinating its attack in the various executions of the protocol.

For $T \in \{\texttt{sequential}, \texttt{parallel}, \texttt{concurrent}\}$, we say that a protocol is $T$-zero-knowledge if it is zero-knowledge under a composition of type $T$. The definitions of $T$-zero-knowledge are derived from Definition 4 by considering appropriate adversaries (i.e., adversarial verifiers); that is, adversaries that can initiate a polynomial number of interactions with the prover, where these interactions are scheduled according to the type $T$.[13] The corresponding simulator (which, as usual, interacts with nobody) is required to produce an output that is computationally indistinguishable from the output of such a type $T$ adversary.

### 5.1 Sequential Composition

In this case, the protocol is invoked (polynomially) many times, where each invocation follows the termination of the previous one. At the very least, security (e.g., zero-knowledge) should be preserved under sequential composition, or else the applicability of the protocol is highly limited (because one cannot safely use it more than once).

Referring to Definition 4, we mention that whereas the "simplified" version (i.e., without auxiliary inputs) is not closed under sequential composition (cf. [55]), the actual version (i.e., with auxiliary inputs) is closed under sequential composition (cf. [59]). We comment that the same phenomena arises when trying to use a zero-knowledge proof as a sub-protocol inside larger protocols. Indeed, it is for these reasons that the augmentation of the "most basic" definition by auxiliary inputs was adopted in all subsequent works.[14]

**Bottom-line:** Every protocol that is zero-knowledge (under Definition 4) is sequential-zero-knowledge.

### 5.2 Parallel Composition

In this case, (polynomially) many instances of the protocol are invoked at the same time and proceed at the same pace. That is, we assume a synchronous model of communication, and consider (polynomially) many executions that are totally synchronized so that the $i$th message in all instances is sent exactly (or approximately) at the same time. (Natural variants on this model are discussed below as well as at the end of Section 5.3.)

It turns out that, in general, zero-knowledge is not closed under parallel composition. A simple counter-example (to the "parallel composition conjecture") is depicted in Figure 2. This counter-example, which is adapted from [55], consists of a simple protocol that is zero-knowledge (in a strong sense), but is not closed under parallel composition (not even in a very weak sense).

We comment that, at the 1980's, the study of parallel composition was interpreted mainly in the context of *round-efficient error reduction* (cf. [39, 55]); that is, the construction of full-fledge zero-knowledge proofs (of negligible soundness error) by composing (in parallel) a basic zero-knowledge protocol of high (but bounded away from 1) soundness error. Since alternative ways of constructing constant-round zero-knowledge proofs (and arguments) were found (cf. [54, 40, 23]), interest in parallel composition (of zero-knowledge protocols) has died. In retrospect, this was a conceptual mistake, because parallel composition (and mild extensions of this notion) capture the preservation of security in a fully synchronous (or almost-fully synchronous) communication network. We note that the almost-fully synchronous communication model is quite realistic in many settings, although it is certainly preferable not to assume even weak synchronism.

[13] Without loss of generality, we may assume that the adversary never violates the scheduling condition; it may instead send an illegal message at the latest possible adequate time. Furthermore, without loss of generality, we may assume that all the adversary's messages are delivered at the latest possible adequate time.

[14] Interestingly, the preliminary version of Goldwasser, Micali and Rackoff's work [66] used the "most basic" definition, whereas the final version of this work used the augmented definition. In some works, the "most basic" definition is used for simplicity, but typically one actually needs the augmented definition.

Consider a party $P$ holding a random (or rather pseudorandom) function $f : \{0,1\}^{2n} \to \{0,1\}^n$, and willing to participate in the following protocol (with respect to security parameter $n$). The other party, called $A$ for adversary, is supposed to send $P$ a binary value $v \in \{1,2\}$ specifying which of the following cases to execute:

For $v = 1$: Party $P$ uniformly selects $\alpha \in \{0,1\}^n$, and sends it to $A$, which is supposed to reply with a pair of $n$-bit long strings, denoted $(\beta, \gamma)$. Party $P$ checks whether or not $f(\alpha\beta) = \gamma$. In case equality holds, $P$ sends $A$ some secret information.

For $v = 2$: Party $A$ is supposed to uniformly select $\alpha \in \{0,1\}^n$, and sends it to $P$, which selects uniformly $\beta \in \{0,1\}^n$, and replies with the pair $(\beta, f(\alpha\beta))$.

Observe that $P$'s strategy is zero-knowledge (even w.r.t auxiliary-inputs as defined in Definition 4): Intuitively, if the adversary $A$ chooses the case $v = 1$, then it is infeasible for $A$ to guess a passing pair $(\beta, \gamma)$ with respect to a random $\alpha$ selected by $P$. Thus, except with negligible probability (when it may get secret information), $A$ does not obtain anything from the interaction. On the other hand, if the adversary $A$ chooses the case $v = 2$, then it obtains a pair that is indistinguishable from a uniformly selected pair of $n$-bit long strings (because $\beta$ is selected uniformly by $P$, and for any $\alpha$ the value $f(\alpha\beta)$ looks random to $A$).

In contrast, if the adversary $A$ can conduct two concurrent[a] executions with $P$, then it may learn the desired secret information: In one session, $A$ sends $v = 1$ while in the other it sends $v = 2$. Upon receiving $P$'s message, denoted $\alpha$, in the first session, $A$ sends it as its own message in the second session, obtaining a pair $(\beta, f(\alpha\beta))$ from $P$'s execution of the second session. Now, $A$ sends the pair $(\beta, f(\alpha\beta))$ to the first session of $P$, this pair passes the check, and so $A$ obtains the desired secret.

[a] Dummy messages may be added (in both cases) in order to obtain the above scheduling in the perfectly parallel case.

Figure 2: A counter-example (adapted from [55]) to the parallel repetition conjecture for zero-knowledge protocols.

Although, in general, zero-knowledge is not closed under parallel composition, under standard intractability assumptions (e.g., the intractability of factoring), there exists zero-knowledge proofs for $\mathcal{NP}$ that are closed under parallel composition. Furthermore, these protocols have a constant number of rounds (cf. [51] for proofs and [34] for arguments).[15] Both results extend also to concurrent composition in a synchronous communication model, where the extension is in allowing protocol invocations to start at different (synchronous) times (and in particular executions may overlap but not run simultaneously).

We comment that parallel composition is problematic also in the context of reducing the soundness error of arguments (cf. [14]), but our focus here is on the zero-knowledge aspect of protocols regardless if they are proofs, arguments or neither.

**Bottom-line:** Under standard intractability assumptions, every NP-set has a constant-round parallel-zero-knowledge proof.

### 5.3 Concurrent Composition (with and without timing)

Concurrent composition generalizes both sequential and parallel composition. Here (polynomially) many instances of the protocol are invoked at arbitrary times and proceed at arbitrary pace. That is, we assume an asynchronous (rather than synchronous) model of communication.

[15] In case of parallel-zero-knowledge *proofs*, there is no need to specify the soundness error because it can always be reduced via parallel composition. As mentioned above, this is not the case with respect to arguments.

In the 1990's, when extensive two-party (and multi-party) computations became a reality (rather than a vision), it became clear that it is (at least) desirable that cryptographic protocols maintain their security under concurrent composition (cf. [33]). In the context of zero-knowledge, concurrent composition was first considered by Dwork, Naor and Sahai [34]. Actually, two models of concurrent composition were considered in the literature, depending on the underlying model of communication (i.e., a *purely asynchronous model* and an *asynchronous model with timing*). Both models cover sequential and parallel composition as special cases.

**Concurrent composition in the pure asynchronous model.** Here we refer to the standard model of asynchronous communication. In comparison to the timing model, the pure asynchronous model is a simpler model and using it requires no assumptions about the underlying communication channels, but it seems harder to construct concurrent zero-knowledge protocols for this model. In particular, for a while it was not known whether concurrent zero-knowledge proofs for $\mathcal{NP}$ exist at all (in this model). Under standard intractability assumptions (e.g., the intractability of factoring), this question was affirmatively resolved by Richardson and Kilian [81]. Following their work, research has focused on determining the round-complexity of concurrent zero-knowledge proofs for $\mathcal{NP}$. This question is still opened, and the current state of the art regarding it is as follows:

- Under standard intractability assumptions, every language in $\mathcal{NP}$ has a concurrent zero-knowledge proof with *almost-logarithmically* many rounds (cf. [80], building upon [72], which in turn builds over [81]). Furthermore, the zero-knowledge property can be demonstrated using a black-box simulator (see definition in Sections 3.3.1 and 6).
- Black-box simulator cannot demonstrated the concurrent zero-knowledge property of non-trivial proofs (or arguments) having significantly less than logarithmically-many rounds (cf. Canetti *et. al.* [28]).[16]
- Recently, Barak [7] demonstrated that the "black-box simulation barrier" can be bypassed. With respect to concurrent zero-knowledge he only obtains partial results: constant-round zero-knowledge arguments (rather than proofs) for $\mathcal{NP}$ that maintain security as long as an a-priori bounded (polynomial) number of executions take place concurrently. (The length of the messages in his protocol grows linearly with this a-priori bound.)

Thus, it is currently unknown whether or not *constant-round* arguments for $\mathcal{NP}$ may be concurrent zero-knowledge (in the pure asynchronous model).

**Concurrent composition under the timing model:** A model of naturally-limited synchronousness (which certainly covers the case of parallel composition) was introduced by Dwork, Naor and Sahai [34]. Essentially, they assume that each party holds a local clock such that the relative clock rates are bounded by an a-priori known constant, and consider protocols that employ time-driven operations (i.e., `time-out` in-coming messages and `delay` out-going messages). The benefit of the timing model is that it is known to construct concurrent zero-knowledge protocols for it. Specifically, using standard intractability assumptions, *constant-round* arguments and proofs that are concurrent zero-knowledge under the timing model do exist (cf. [34] and [51], respectively). The disadvantages of the timing model are discussed next.

[16] By *non-trivial* proof systems we mean ones for languages outside $\mathcal{BPP}$, whereas by *significantly less than logarithmic* we mean any function $f:\mathbb{N}\to\mathbb{N}$ satisfying $f(n) = o(\log n/\log\log n)$. In contrast, by *almost-logarithmically* we mean any function $f$ satisfying $f(n) = \omega(\log n)$.

The timing model consists of the *assumption* that talking about the actual timing of events is meaningful (at least in a weak sense) and of the *introduction of time-driven operations.* The timing assumption amounts to postulating that each party holds a local clock and knows a global bound, denoted $\rho \geq 1$, on the relative rates of the local clocks.[17] Furthermore, it is postulated that the parties know a (pessimistic) bound, denoted $\Delta$, on the message-delivery time (which also includes the local computation and handling times). In our opinion, these timing assumptions are most reasonable, and are unlikely to restrict the scope of applications for which concurrent zero-knowledge is relevant. We are more concerned about the effect of the time-driven operations introduced in the timing model. Recall that these operations are the `time-out` of in-coming messages and the `delay` of out-going messages. Furthermore, typically the delay period is at least as long as the time-out period, which in turn is at least $\Delta$ (i.e., the time-out period must be at least as long as the pessimistic bound on message-delivery time so not to disrupt the proper operation of the protocol). This means that the use of these time-driven operations yields slowing down the execution of the protocol (i.e., running it at the rate of the pessimistic message-delivery time rather than at the rate of the actual message-delivery time, which is typically much faster). Still, in absence of more appealing alternatives (i.e., a constant-round concurrent zero-knowledge protocol for the pure asynchronous model), the use of the timing model may be considered reasonable. (We comment than other alternatives to the timing-model include various set-up assumptions; cf. [26, 30].)

**Back to parallel composition:** Given our opinion about the timing model, it is not surprising that we consider the problem of parallel composition almost as important as the problem of concurrent composition in the timing model. Firstly, it is quite reasonable to assume that the parties' local clocks have approximately the same rate, and that drifting is corrected by occasional clock synchronization. Thus, it is reasonable to assume that the parties have approximately-good estimate of some global time. Furthermore, the global time may be partitioned into phases, each consisting of a constant number of rounds, so that each party wishing to execute the protocol just delays its invocation to the beginning of the next phase. Thus, concurrent execution of (constant-round) protocols in this setting amounts to a sequence of (time-disjoint) almost-parallel executions of the protocol. Consequently, proving that the protocol is parallel zero-knowledge suffices for concurrent composition in this setting.

**Relation to resettable zero-knowledge.** Going to the other extreme, we mention that there is a model of zero-knowledge that is even stronger than concurrent zero-knowledge (even in the pure asynchronous model). Specifically, "resettable zero-knowledge" as defined in Section 11, implies concurrent zero-knowledge.

## 6 Using the adversary's program in the proof of security

As discussed in the first part of this tutorial, zero-knowledge is defined by following the simulation paradigm, which in turn underlies many other central definitions in cryptography. Recall that the definition of zero-knowledge proofs states that whatever an efficient adversary can compute after interacting with the prover, can actually be efficiently computed from scratch by a so-called *simulator* (which works without interacting with the prover). Although the simulator may depend arbitrarily on the adversary, the need to present a simulator for each feasible adversary seems to

[17]The rate should be computed with respect to reasonable intervals of time; for example, for $\Delta$ as defined below, one may assume that a time period of $\Delta$ units is measured as $\Delta'$ units of time on the local clock, where $\Delta/\rho \leq \Delta' \leq \rho\Delta$.

require the presentation of a universal simulator that is given the adversary's strategy (or program) as another auxiliary input. The question addressed in this section is how can the universal simulator use the adversary's program.

The adversary's program (or strategy) is actually a function determining for each possible view of the adversary (i.e., its input, random choices and the message it has received so far) which message will be sent next. Thus, we identify the adversary's program with this next-message function. As stated in Section 3.3.1, until very recently, all universal simulators (constructed towards demonstrating zero-knowledge properties) have used the adversary's program (or rather its next-message function) as a black-box (i.e., the simulator invoked the next-message function on a sequence of arguments of its choice). Furthermore, in view of the presumed difficulty of "reverse engineering" programs, it was commonly believed that nothing is lost by restricting attention to simulators, called black-box simulators, that only make black-box usage of the adversary's program. Consequently, Goldreich and Krawczyk conjectured that impossibility results regarding black-box simulation represent inherent limitations of zero-knowledge itself, and studied the limitations of the former [55].

> In particular, they showed that parallel composition of the protocol of Figure 1 (as well as of any constant-round public-coin protocol) *cannot be proven to be zero-knowledge using a black-box simulator*, unless the language (i.e., 3-Colorability) is in $\mathcal{BPP}$. In fact their result refers to any constant-round public-coin protocol with negligible soundness error, regardless of how such a protocol is obtained. This result was taken as strong evidence towards the conjecture that constant-round public-coin protocol with negligible soundness error *cannot be zero-knowledge* (unless the language is in $\mathcal{BPP}$).
>
> Similarly, as mentioned in Section 5.3, it was shown that protocols of sub-logarithmic number of rounds *cannot be proven to be concurrent zero-knowledge via a black-box simulator* [28], and this was taken as evidence towards the conjecture that such protocols cannot be *concurrent zero-knowledge*.

In contrast to these conjectures and supportive evidence, Barak showed how to constructed non-black-box simulators and obtained several results that were known to be unachievable via black-box simulators [7]. In particular, under standard intractability assumption (see also [9]), he presented constant-round public-coin zero-knowledge arguments with negligible soundness error for any language in $\mathcal{NP}$. (Moreover, the simulator runs in strict polynomial-time, which is impossible for black-box simulators of non-trivial constant-round protocols [11].) Furthermore, this protocol preserves zero-knowledge under a fixed[18] polynomial number of concurrent executions, in contrast to the result of [28] (regarding black-box simulators) that holds also in that restricted case. Thus, Barak's result calls for the re-evaluation of many common believes. Most concretely, it says that results regarding black-box simulators do not reflect inherent limitations of zero-knowledge (but rather an inherent limitation of a natural way of demonstrating the zero-knowledge property). Most abstractly, it says that there are meaningful ways of using a program other than merely invoking it as a black-box.

Does this means that a method was found to "reverse engineer" programs or to "understand" them? We believe that the answer is negative. Barak [7] is using the adversary's program in a significant way (i.e., more significant than just invoking it), without "understanding" it. *So how does he use the program?*

[18]The protocol depends on the polynomial bounding the number of executions, and thus is not known to be concurrent zero-knowledge (because the latter requires to fix the protocol and then consider any polynomial number of concurrent executions).

The key idea underlying Barak's argument system [7] is to have the prover prove that either the original NP-assertion is valid or that he (i.e., the prover) "knows the verifier's residual strategy" (in the sense that it can predict the next verifier message). Indeed, in a real interaction (with the honest verifier), it is infeasible for the prover to predict the next verifier message, and so computational-soundness of the protocol follows. However, a simulator that is given the code of the verifier's strategy (and not merely oracle access to that code), can produce a valid proof of the disjunction by properly executing the sub-protocol using its knowledge of an NP-witness for the second disjunctive. The simulation is computational indistinguishable from the real execution, provided that one cannot distinguish an execution of the sub-protocol in which one NP-witness (i.e., an NP-witness for the original assertion) is used from an execution in which the second NP-witness (i.e., an NP-witness for the auxiliary assertion) is use. That is, the sub-protocol should be a *witness indistinguishable* argument system (see further discussion below). We warn the reader that the actual implementation of the above idea requires overcoming several technical difficulties (cf. [7, 9]).

**Perspective.** In retrospect, taking a wide perspective, it should not come as a surprise that the program's code yields extra power beyond black-box access to it. Feeding a program with its own code (or part of it) is the essence of the diagonalization technique, and this too is done without "reverse engineering". Furthermore, various non-black-box techniques have appeared before in the cryptographic setting, but they were used in the more natural context of devising an attack on an (artificial) insecure scheme (e.g., towards proving the failure of the "Random Oracle Methodology" [27] and the impossibility of software obfuscation [10]). In contrast, in [7] (and [8]) the code of the adversary is being used within a sophisticated proof of security. What we wish to highlight here is that non-black-box usage of programs is relevant also to proving (rather than to disproving) the security of systems.

### Digest: Witness Indistinguishability and the FLS-Technique

The above description (of [7]), as well as several other sophisticated constructions of zero-knowledge protocols (e.g., [38, 81]), makes crucial use of a technique introduced by Feige, Lapidot and Shamir [38], which in turn is based on the notion of witness indistinguishability (introduced by Feige and Shamir [39]).

Loosely speaking, for any NP-relation $R$, an argument system for the corresponding language (i.e., $L_R$) is called witness indistinguishable if no feasible verifier may distinguish the case in which the prover uses one NP-witness to $x$ (i.e., $w_1$ such that $(x, w_1) \in R$) from the case the prover is using a different NP-witness to the same input $x$ (i.e., $w_2$ such that $(x, w_2) \in R$). Furthermore, if $x_1$ is indistinguishable from $x_2$ then no feasible verifier may distinguish the case in which the prover uses $w_1$ to prove $x_1 \in L_R$ from the case that the prover uses $w_2$ to prove $x_2 \in L_R$.[19] Clearly, any zero-knowledge protocol is witness indistinguishable, but the converse does not necessarily hold and it seems that witness indistinguishable protocols are easier to construct than zero-knowledge ones. (We mention that witness indistinguishable protocols are closed under parallel composition [39], whereas this does not hold in general for zero-knowledge protocols.)

Following is a sketchy description of a special case of the FLS-technique, whereas the above-mentioned application uses a more general version (which refers to proofs of knowledge, as defined

[19] The additional condition yields a stronger notion (cf. [49, Def. 4.6.2]), but for simplicity we call it witness indistinguishability.

in Section 7).[20] In this special case, the technique consists of the following construction schema, which uses witness indistinguishable protocols for $\mathcal{NP}$ in order to obtain zero-knowledge protocols for $\mathcal{NP}$. On common input $x \in L$, where $L = L_R$ is the NP-set defined by the witness relation $R$, the following two steps are performed:

1. The parties generate an instance $x'$ for an auxiliary NP-set $L'$, where $L'$ is defined by the witness relation $R'$. The generation protocol in use must satisfy two conditions:
   (a) If the verifier follows its prescribed strategy then no matter which feasible strategy is used by the prover, with high probability, the outcome $x'$ is a NO-instance of $L'$.
   (b) Loosely speaking, it is feasible to generate a transcript of the generation protocol that is computationally indistinguishable from the real interaction *along with an NP-witness for the outcome of the protocol.*
2. The parties execute a witness indistinguishable protocol for the set $L''$ defined by the witness relation $R'' = \{((y, y'), (z, z')) : (y, z) \in R \vee (y', z') \in R'\}$. The sub-protocol is such that the corresponding prover can be implemented in probabilistic polynomial-time given an NP-witness for $(y, y') \in L''$. The sub-protocol is invoked on common input $(x, x')$, where $x'$ is the outcome of Step 1, and the sub-prover is invoked with the corresponding NP-witness as auxiliary input (i.e., with $(w, \lambda)$, where $w$ is the NP-witness for $x$ given to the main prover).

The computational-soundness of the above protocol follows by Property (a) of the generation protocol (i.e., with high probability $x' \notin L'$, and so $x \in L$ by the soundness of the protocol used in Step 2). To demonstrate the zero-knowledge property, we first generate a simulated transcript of Step 1 (with outcome $x' \in L'$) along with an adequate NP-witness (i.e., $w'$ such that $(x', w') \in L'$), and then emulates Step 2 by feeding the sub-prover strategy with the NP-witness $(\lambda, w')$. Combining Property (b) of the generation protocol and the witness indistinguishability property of the protocol used in Step 2, the simulation is indistinguishable from the real execution.

# 7 Proof of Knowledge

This section addresses the concept of "proofs of knowledge". Loosely speaking, these are proofs in which the prover asserts "knowledge" of some object (e.g., a 3-coloring of a graph), and not merely its existence (e.g., the existence of a 3-coloring of the graph, which in turn implies that the graph is 3-colorable). But what is meant by saying that a machine knows something? Indeed the main thrust of this section is in addressing this question. Before doing so we point out that "proofs of knowledge", and in particular zero-knowledge "proofs of knowledge", have many applications to the design of cryptographic schemes and cryptographic protocols. In fact, we have already referred to "proofs of knowledge" in Section 6.

## 7.1 How to define proofs of knowledge

What does it mean to say that a *machine* knows something? Any standard dictionary suggests several meanings for the verb `to know`, and most meanings are phrased with reference to *awareness*,

[20] In the general case, the generation protocol may generate instances $x'$ in $L'$, but it is infeasible for the prover to obtain a corresponding witness (i.e., a $w'$ such that $(x', w') \in R'$). In the second step, the sub-protocol in use ought to be a proof of knowledge, and computational-soundness of the main protocol will follows (because otherwise the prover, using a knowledge extractor, can obtain a witness for $x' \in L'$).

a notion which is certainly inapplicable in the context of machines. We must look for a *behavioristic* interpretation of the verb `to know`. Indeed, it is reasonable to link knowledge with ability to do something (e.g., the ability to write down whatever one knows). Hence, we will say that a machine knows a string $\alpha$ if it *can* output the string $\alpha$. But this seems as total non-sense too: a machine has a well defined output – either the output equals $\alpha$ or it does not. *So what can be meant by saying that a machine can do something?* Loosely speaking, it may mean that the machine can be *easily modified* so that it does whatever is claimed. More precisely, it may mean that there exists an *efficient* machine that, using the original machine as a black-box (or given its code as an input), outputs whatever is claimed.

So much for defining the "knowledge of machines". Yet, whatever a machine knows or does not know is "its own business". What can be of interest and reference *to the outside* is the question of what can be deduced about the knowledge of a machine after interacting with it. Hence, we are interested in proofs of knowledge (rather than in mere knowledge).

For sake of simplicity let us consider a concrete question: *how can a machine prove that it knows a 3-coloring of a graph?* An obvious way is just to send the 3-coloring to the verifier. Yet, we claim that applying protocol in Figure 1 (i.e., the zero-knowledge proof system for 3-Colorability) is an alternative way of proving knowledge of a 3-coloring of the graph.

Loosely speaking, we may say that an interactive machine, $V$, constitutes a verifier for knowledge of 3-coloring if the probability that the verifier is convinced by a machine $P$ to accept the graph $G$ is inversely proportional to the difficulty of extracting a 3-coloring of $G$ when using machine $P$ as a "black box".[21] Namely, the extraction of the 3-coloring is done by an oracle machine, called an *extractor*, that is given access to a function specifying the behavior $P$ (i.e., the messages it sends in response to particular messages it may receives). We require that the (expected) running time of the extractor, on input $G$ and access to an oracle specifying $P$'s messages, be inversely related (by a factor polynomial in $|G|$) to the probability that $P$ convinces $V$ to accept $G$. In case $P$ always convinces $V$ to accept $G$, the extractor runs in expected polynomial-time. The same holds in case $P$ convinces $V$ to accept with noticeable probability. (We stress that the latter special cases do not suffice for a satisfactory definition; see discussion in [49, Sec. 4.7.1].)[22]

We mention that the concept of proofs of knowledge was first introduced in [66], but the above formulation is based mostly on [13]. A famous application of zero-knowledge proofs of knowledge is to the construction of identification schemes (e.g., the Fiat-Shamir scheme [41]).

## 7.2 How to construct proofs of knowledge

As hinted above, many of the known proof systems are in fact proofs of knowledge. Furthermore, some (but not all) known zero-knowledge proofs (resp., arguments) are in fact proofs (resp., arguments) of knowledge.[23] Indeed, a notable example is the zero-knowledge proof depicted in Figure 1. For further discussion, see [49, Sec. 4.7] and [11].

[21] Indeed, as hinted above, one may consider also non-black-box extractors as done in [11]. However, this limits the applicability of the definitions to provers that are implemented by polynomial-size circuits.

[22] In particular, note that the latter probability may be neither noticeable (i.e., bounded below by the reciprocal of some polynomial) nor negligible (i.e., bounded above by the reciprocal of every polynomial). This means that we may neither see the event if we make an a-priori bounded polynomial number of trials nor ignore it.

[23] Arguments of knowledge are defined analogous to proofs of knowledge, while limiting the extraction requirement to provers that are implemented by polynomial-size circuits. In this case, it is natural to allow also non-black-box extraction, as discussed in Footnote 21.

# 8 Non-Interactive Zero-Knowledge

In this section we consider non-interactive zero-knowledge proof systems. The model, introduced in [18], consists of three entities: a prover, a verifier and a uniformly selected reference string (which can be thought of as being selected by a trusted third party). Both verifier and prover can read the reference string, and each can toss additional coins. The interaction consists of a single message sent from the prover to the verifier, who then is left with the final decision (whether to accept or not). The (basic) zero-knowledge requirement refers to a simulator that outputs pairs that should be computationally indistinguishable from the distribution (of pairs consisting of a uniformly selected reference string and a random prover message) seen in the real model.[24] Non-interactive zero-knowledge proof systems have numerous applications (e.g., to the construction of public-key encryption and signature schemes, where the reference string may be incorporated in the public-key). Several different definitions of non-interactive zero-knowledge proofs were considered in the literature.

- In the *basic definition*, one considers proving a single assertion of a-priori bounded length, where this length may be smaller than the length of the reference string.
- A natural extension, required in many applications, is the ability to prove multiple assertions of varying length, where the total length of these assertions may exceed the length of the reference string (as long as the total length is polynomial in the length of the reference string). This definition is sometimes referred to as the *unbounded definition*, because the total length of the assertions to be proven is not a-priori bounded.
- Other natural extensions refer to the preservation of security (i.e., both soundness and zero-knowledge) when the assertions to be proven are selected *adaptivity* (based on the reference string and possibly even based on previous proofs).
- Finally, we mention the notion of *simulation-soundness*, which is related to *non-malleability*. This extension, which mixes the zero-knowledge and soundness conditions, refers to the soundness of proofs presented by an adversary after it obtains proofs of assertions of its own choice (with respect to the same reference string). This notion is important in applications of non-interactive zero-knowledge proofs to the construction of public-key encryption schemes secure against chosen ciphertext attacks (see [50, Sec. 5.4.4.4]).

Constructing non-interactive zero-knowledge proofs seems more difficult than constructing interactive zero-knowledge proofs. Still, based on standard intractability assumptions (e.g., intractability of factoring), it is known how to construct a non-interactive zero-knowledge proof (even in the adaptive and non-malleable sense) for any NP-set.

**Suggestions for further reading:** For a definitional treatment of the basic, unbounded and adaptive definitions see [49, Sec. 4.10]. Increasingly stronger variants of the non-malleable definition are presented in [50, Sec. 5.4.4.4] and [31]. A relatively simple construction for the basic model is presented in [38] (see also [49, Sec. 4.10.2]). (A more efficient construction can be found in [71].) See [38] (or [49, Sec. 4.10.3]) for a transformation of systems for the basic model into systems for the unbounded model. Constructions for increasingly stronger variants of the (adaptive) non-malleable definition are presented in [50, Sec. 5.4.4.4] and [31].

[24] Note that the verifier does not effect the distribution seen in the real model, and so the basic definition of zero-knowledge does not refer to it. The verifier (or rather a process of adaptively selecting assertions to be proven) will be referred to in the adaptive variants of the definition.

# 9 Statistical Zero-Knowledge

Recall that statistical zero-knowledge protocols are such in which the distribution ensembles referred to in Definition 4 are required to be *statistically indistinguishable* (rather than *computationally indistinguishable*). Under standard intractability assumptions, every NP-set has a *statistical* zero-knowledge *argument* [21]. On the other hand, it is unlikely that all NP-sets have *statistical* zero-knowledge *proofs* [42, 1]. Currently, the intractability assumption used for constructing *statistical* zero-knowledge *arguments* (for $\mathcal{NP}$) seems stronger than the assumption used for constructing *computational* zero-knowledge *proofs* (for $\mathcal{NP}$). Assuming both constructs exist, the question of which to prefer depends on the application (e.g., is it more important to protect the prover's secrets or to protect the verifier from being convinced of false assertions). Statistical zero-knowledge proofs, whenever they exist, free us from this dilemma. Indeed, this is one out of several reasons for studying these objects. That is:

- Statistical zero-knowledge proofs offer information-theoretic security to both parties. Thus, whenever they exist, statistical zero-knowledge proofs may be preferred over computational zero-knowledge proofs (which only offer computational security to the prover) and over statistical zero-knowledge arguments (which only offer computational security to the verifier).
- Statistical zero-knowledge proofs provide a clean model for the study of various properties, which may result in techniques that are applicable also for computational zero-knowledge. (One example is mentioned below.)
- The class of problems having statistical zero-knowledge proofs is interesting from a complexity theoretic point of view. On one hand, this class is likely to be a proper superset of $\mathcal{BPP}$ (e.g., it contains seemingly hard problems such as Quadratic Resideousity [66], Graph Isomorphism [57], and a promise problem equivalent to the Discrete Logarithm Problem [56]). On the other hand, this class is contained in $\mathcal{AM} \cap \mathrm{co}\mathcal{AM}$ (cf. [1, 42]), which is believed not to extend much beyond $\mathcal{NP} \cap \mathrm{co}\mathcal{NP}$. ($\mathcal{AM}$ is the class of sets having two-round public-coin interactive proofs.)

In the rest of this section, we survey the main results regarding the internal structure of the class of sets having statistical zero-knowledge proofs. This study was initiated to a large extent by Okamoto [77]. We first present *transformations* that, when applied to certain statistical zero-knowledge protocols, yield protocols with additional properties. Next, we consider several structural properties of the class, most notably the existence of *natural* complete problems (discovered by Sahai and Vadhan [83]).

## 9.1 Transformations

The first transformation takes any public-coin interactive proof that is statistical zero-knowledge with respect to the honest verifier, and returns a (public-coin) statistical zero-knowledge [62]. When applied to a public-coin interactive proof that is (computational) zero-knowledge with respect to the honest verifier, the transformation yields a (computational) zero-knowledge proof. Thus, this transformation "amplifies the security" of (public-coin) protocols, from leaking nothing to the prescribed verifier into leaking nothing to any cheating verifier.

The heart of the transformation is a suitable random selection protocol, which is used to emulate the verifier's messages in the original protocol. Loosely speaking, the random selection protocol is zero-knowledge in a strong sense, and the effect of each of the parties on the protocol's outcome is adequately bounded. For example, it is impossible for the verifier to effect the protocol's outcome (by more than a negligible amount), whereas the prover cannot increase the probability that the outcome hits any set by more than some specific (super-polynomial) factor.

The first transformation calls our attention to public-coin interactive proofs that are statistical zero-knowledge (with respect to the honest verifier). In general, public-coin interactive proofs are easier to manipulate than general interactive proofs. The second transformation takes any are statistical zero-knowledge (with respect to the honest verifier) proof and returns one that is of the public-coin type (see [64], which builds on [77]). Unfortunately, the second transformation, which is analogous to a prior result regarding interactive proofs [67], does *not* extend to computational zero-knowledge,

Combined together, the two transformations imply that the class of sets (or promise problems) having interactive proofs that are statistical zero-knowledge with respect to the honest verifier equals the class of sets having (general) statistical zero-knowledge proofs.

### 9.2 Complete problems and structural properties

In the rest of this section we consider classes of *promise problems* (rather than classes of decision problems or sets). Specifically, we denote by $\mathcal{SZK}$ the class of problems having a statistical zero-knowledge proof.

One remarkable property of the class $\mathcal{SZK}$ is that it has natural complete problems (i.e., problems in $\mathcal{SZK}$ such that any problem in $\mathcal{SZK}$ is Karp-reducible to them). One such problem is to distinguish pairs of distributions (given via sampling circuits) that are statistically close from pairs that are statistically far apart [83]. Another such problem is, given two distributions of sufficiently different entropy, to tell which has higher entropy [64]. It is indeed interesting that "the class statistical zero-knowledge is all about statistics (or probability)".

Another remarkable property of $\mathcal{SZK}$ is the fact that it is closed under complementation (see [83], which builds on [77]). In fact, $\mathcal{SZK}$ is closed under $\mathcal{NC}_1$-truth-table reductions [83]. Recall that $\mathcal{BPP} \subseteq \mathcal{SZK} \subseteq \mathcal{AM} \cap \mathrm{co}\mathcal{AM}$, and that the first inclusion is believed to be strict.

**Non-Interactive SZK.** A systematic study of Non-Interactive Statistical Zero-Knowledge proof systems was conducted in [63]. The main result is evidence to the non-triviality of the class (i.e., it contains sets outside $\mathcal{BPP}$ if and only if $\mathcal{SZK} \neq \mathcal{BPP}$).

## 10 Knowledge Complexity

One of the many contributions of the seminal paper of Goldwasser, Micali and Rackoff [66] is the introduction of the concept of knowledge complexity. Knowledge complexity is intended to measure the computational advantage gained by interaction. Hence, something that can be obtained without interaction is not considered knowledge. The latter phrase is somewhat qualitative and supplies the intuition underlying the definition of zero-knowledge (i.e., knowledge complexity zero) as surveyed above. Quantifying the amount of knowledge gained by interaction, in case it is not zero, is more problematic.[25] We stress that the definition of zero-knowledge *does not* depend on the formulation of *the* AMOUNT *of knowledge gained* since this definition addresses the case in which NO *knowledge is gained.*

Several definitions of knowledge complexity has appeared in the literature, where some are closely related and quite robust (cf. [61]). Here we survey one definitional approach, which we

[25]It seems that, in general, quantitative notions are harder to handle than qualitative ones.

consider most satisfactory. According to this approach the amount of knowledge gained in an interaction is bounded by the number of bits that are communicated in an alternative interaction that allows to simulate the original interaction. That is, party $P$ is said to yield at most $k$ bits of knowledge (on inputs in $S$) if whatever can be efficiently computed through an interaction with $P$ on common input $x \in S$, can also be efficiently computed through an interaction (on the same common input $x$) with an alternative machine $P'$ that sends at most $k(|x|)$ bits. This formulation can be applied with respect to various types of simulations, extending the various types of zero-knowledge. Our focus is on the extension of *statistical* zero-knowledge proofs (because, under standard intractability assumptions, any language in $\mathcal{IP}$ has a computational zero-knowledge proof). We note that, without loss of generality, the "knowledge-giving-machine" can be made memoryless and deterministic (i.e., by providing it with all previous messages and with coin tosses). Hence, the "knowledge-giving-machine" is merely an oracle (and we may think of the simulation as being performed by an oracle machine and count the number of its binary queries). For further discussion of this and other definitions, the reader is referred to [61].

A natural research project is to characterize languages according to the (statistical) knowledge-complexity of their interactive proof systems. The main result known (for the above definition) is that languages with logarithmic statistical knowledge-complexity are in $\mathcal{AM} \cap \mathrm{co}\mathcal{AM}$ (cf. [79], building on [1] and [60]). Thus, unless the polynomial time hierarchy collapses (cf. [20]), NP-complete set have super-logarithmic statistical knowledge-complexity.

## 11 Resettability of a party's random-tape (rZK and rsZK)

Having gained a reasonable understanding of the security of cryptographic schemes and protocols as stand-alone, cryptographic research is moving towards the study of stronger notions of security. Examples include the effect of executing several instances of the same protocol concurrently (e.g., the malleability of an individual protocol [33]) as well as the effect of executing the protocol concurrently to any other activity (or set of protocols) [25]. Another example of a stronger notion of security, which is of theoretical and practical interest, is the security of protocols under a "resetting" attack. In such an attack a party may be forced to *execute a protocol several times while using the same random-tape* and *without coordinating these executions* (e.g., by maintaining a joint state). The theoretical interest in this notion stems from the fact that randomness plays a pivotal role in cryptography, and thus the question of whether one needs fresh randomness in each invocation of a cryptographic protocol is very natural. The practical importance is due to the fact that in many settings it is impossible or undesirable to generate fresh randomness on the fly.

**Resettable Zero-Knowledge (rZK).** Resettability of players in a cryptographic protocol was first considered in [26], which studies what happens to the security of zero-knowledge interactive proofs and arguments *when the verifier can reset the prover to use the same random tape in multiple concurrent executions.* Protocols that remain zero-knowledge against such a verifier, are called resettable zero-knowledge (rZK). Put differently, the question of prover resettability, is whether zero-knowledge is achievable when the prover cannot use fresh randomness in new interactions, but rather is restricted to (re-)using a fixed number of coins. Resettability implies security under concurrent executions: As shown in [26], any rZK protocol constitutes a concurrent zero-knowledge protocol. The opposite direction does not hold (in general), and indeed it was not a-priori clear whether (non-trivial) rZK protocols may at all exist. Under standard intractability assumptions, it was shown that resettable zero-knowledge interactive proofs for any NP-set do exist [26]. (For related models and efficiency improvements, see [26] and [72], respectively.)

**Resettablly-Sound Zero-Knowledge (rsZK).** Resettably-sound proofs and arguments maintain soundness even *when the prover can reset the verifier to use the same random coins in repeated executions of the protocol*. This notion was studied in [12], who obtained the following results: On one hand, under standard intractability assumptions, any NP-set has a (constant-round) resettably-sound zero-knowledge *argument*. On the other hand, resettably-sound zero-knowledge *proofs* are possible only for languages in $\mathcal{P}$/poly. The question of whether a protocol for $\mathcal{NP}$ can be both resettably-sound and resettably-zero-knowledge is still open.

## 12 Zero-knowledge in other models

As stated above, zero-knowledge is a property of some interactive strategies, regardless of the goal (or other properties) of these strategies. We have seen that zero-knowledge can be meaningfully applied in the context of interactive proofs and arguments. Here we briefly discuss the applicability of zero-knowledge to other settings in which, as in the case of arguments, there are restrictions on the type of prover strategies. We stress that the restrictions discussed here refer to the strategies employed by the prover both in case it tries to prove valid assertions (i.e., the completeness condition) and in case it tries to fool the verifier to believe false statements (i.e., the soundness condition). Thus, the validity of the verifier decision (concerning false statements) depends on whether this restriction (concerning potential "cheating" prover strategies) really holds. The reason to consider these restricted models is that they enable to achieve results that are not possible in the general model of interactive proofs (cf., [16, 21, 70, 75]). We consider restrictions of two types: computational and physical. We start with the latter.

**Multi-Prover Interactive Proofs (MIP).** In the so-called *multi-prover interactive proof* model, denoted MIP (cf., [16]), the prover is split into several (say, two) entities and the restriction (or assumption) is that these entities cannot interact with each other. Actually, the formulation allows them to coordinate their strategies prior to interacting with the verifier[26] but it is crucial that they don't exchange messages among themselves while interacting with the verifier. The multi-prover model is reminiscent of the common police procedure of isolating collaborating suspects and interrogating each of them separately. A typical application in which the two-prover model may be assumed is an ATM that verifies the validity of a pair of smart-cards inserted in two isolated slots of the ATM. The advantage in using such a split system is that it enables the presentation of (perfect) zero-knowledge proof systems for any set in $\mathcal{NP}$, while *using no intractability assumptions* [16].

**Strict Computational-Soundness (a.k.a Timed-ZK).** Recall that we have already discussed one model of computational-soundness; that is, the model of arguments refers to prover strategies that are implementable by probabilistic polynomial-time machines with adequate auxiliary input.[27] A more strict restriction, studied in [35], refers to prover strategies that are implementable within an a-priori fixed number of computation steps (where this number is a fixed polynomial in the length of the common input). In reality, the prover's actual running-time is monitored by the verifier that may run for a longer time, and the prover's utility is due to an auxiliary input that it has. (An analogous model, where the length of the auxiliary input is a-priori fixed, was also considered in [35].)

[26] This is implicit in the universal quantifier used in the soundness condition.

[27] A related model is that of CS-proofs, where the prover's strategy is allowed to run in time that is polynomial in the time it takes to decide membership of the common input via a canonical decision procedure for the language [75].

## 13 A source of inspiration for complexity theory

Throughout the years, zero-knowledge has served as a source of inspiration for models and techniques in complexity theory. The first such case is the very introduction of interactive proofs, which was motivated by the notion of zero-knowledge.

The story begins with Goldwasser, Micali and Rackoff who sought a general setting for their novel notion of zero-knowledge [66]. The choice fell on proof systems as capturing a fundamental activity that takes place in a cryptographic protocol. Motivated by the desire to formulate the most general type of "proofs" that may be used within cryptographic protocols, Goldwasser, Micali and Rackoff introduced the notion of an *interactive proof system* [66]. Although the main focus of their paper is on zero-knowledge, the possibility that interactive proof systems may be more powerful than NP-proof system has been pointed out in [66].

Similarly, the main motivation for the introduction of multi-prover interactive proofs (in [16]) came from zero-knowledge; specifically, introducing multi-prover zero-knowledge proofs for $\mathcal{NP}$ without relying on intractability assumptions. Again, the complexity theoretic prospects of the new class, denoted $\mathcal{MIP}$, have not been ignored. A more appealing, to our taste, formulation of the class $\mathcal{MIP}$ has been subsequently presented in [43]. The latter formulation coincides with the formulation currently known as *probabilistically checkable proofs* (i.e., $\mathcal{PCP}$).

Getting more technical, we mention that the notion of zero-knowledge as well as known zero-knowledge proof systems have inspired constructions that seem unrelated to zero-knowledge. A notable example is the PCP construction of [37], which was tailored towards obtaining tight inapproximability results for the chromatic number.

## References


[1] W. Aiello and J. Håstad. Perfect Zero-Knowledge Languages can be Recognized in Two Rounds. In *28th IEEE Symposium on Foundations of Computer Science*, pages 439–448, 1987.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Intractability of Approximation Problems. *Journal of the ACM*, Vol. 45, pages 501–555, 1998. Preliminary version in *33rd FOCS*, 1992.

[3] S. Arora and S. Safra. Probabilistic Checkable Proofs: A New Characterization of NP. *Journal of the ACM*, Vol. 45, pages 70–122, 1998. Preliminary version in *33rd FOCS*, 1992.

[4] L. Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–420, 1985.

[5] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *23rd ACM Symposium on the Theory of Computing*, pages 21–31, 1991.

[6] L. Babai, L. Fortnow, N. Nisan and A. Wigderson. BPP has Subexponential Time Simulations unless EXPTIME has Publishable Proofs. *Complexity Theory*, Vol. 3, pages 307–318, 1993.

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

[8] B. Barak. Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model. In *43th IEEE Symposium on Foundations of Computer Science*, to appear, 2002.

[9] B. Barak and O. Goldreich, Universal arguments and their applications. In the *17th IEEE Conference on Computational Complexity*, pages 194–203, 2002.

[10] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of software obfuscation. In *Crypto01*, Springer-Verlag Lecture Notes in Computer Science (Vol. 2139), pages 1–18.

[11] B. Barak and Y. Lindell. Non-Black-Box Proofs of Knowledge (tentative title). In *34th ACM Symposium on the Theory of Computing*, pages 484–493, 2002.

[12] B. Barak, O. Goldreich, S. Goldwasser and Y. Lindell, Resettably-Sound Zero-Knowledge and its Applications. In *42th IEEE Symposium on Foundations of Computer Science*, pages 116–125, 2001.

[13] M. Bellare and O. Goldreich. On Defining Proofs of Knowledge. In *Crypto92*, Springer-Verlag Lecture Notes in Computer Science (Vol. 740), pages 390–420.

[14] M. Bellare, R. Impagliazzo and M. Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th IEEE Symposium on Foundations of Computer Science*, pages 374–383, 1997.

[15] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali and P. Rogaway. Everything Provable is Probable in Zero-Knowledge. In *Crypto88*, Springer-Verlag Lecture Notes in Computer Science (Vol. 403), pages 37–56, 1990.

[16] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability. In *20th ACM Symposium on the Theory of Computing*, pages 113–131, 1988.

[17] M. Blum, A. De Santis, S. Micali, and G. Persiano. Non-Interactive Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 20, No. 6, pages 1084–1118, 1991. (Considered the journal version of [18].)

[18] M. Blum, P. Feldman and S. Micali. Non-Interactive Zero-Knowledge and its Applications. In *20th ACM Symposium on the Theory of Computing*, pages 103–112, 1988. See [17].

[19] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, Vol. 13, pages 850–864, 1984. Preliminary version in *23rd FOCS*, 1982.

[20] R. Boppana, J. Håstad, and S. Zachos. Does Co-NP Have Short Interactive Proofs? *Information Processing Letters*, 25, May 1987, pp. 127-132.

[21] G. Brassard, D. Chaum and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Science*, Vol. 37, No. 2, pages 156–189, 1988. Preliminary version by Brassard and Crépeau in *27th FOCS*, 1986.

[22] G. Brassard and C. Crépeau. Zero-Knowledge Simulation of Boolean Circuits. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 223–233, 1987.

[23] G. Brassard, C. Crépeau and M. Yung. Constant-Round Perfect Zero-Knowledge Computationally Convincing Protocols. *Theoretical Computer Science*, Vol. 84, pages 23–52, 1991.

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

[25] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001. Full version (with different title) is available from *Cryptology ePrint Archive*, Report 2000/067.

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

[27] R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. In *30th ACM Symposium on the Theory of Computing*, pages 209–218, 1998. Full version available on-line from `http://eprint.iacr.org/1998/011`.

[28] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-Box Concurrent Zero-Knowledge Requires $\tilde{\Omega}(\log n)$ Rounds. In *33rd ACM Symposium on the Theory of Computing*, pages 570–579, 2001.

[29] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *34th ACM Symposium on the Theory of Computing*, pages 494–503, 2002.

[30] I. Damgård. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In *Eurocrypt'00*, 2000.

[31] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust Non-interactive Zero-Knowledge. In *Crypto01*, Springer Lecture Notes in Computer Science (Vol. 2139), pages 566–598.

[32] W. Diffie, and M.E. Hellman. New Directions in Cryptography. *IEEE Trans. on Info. Theory*, IT-22 (Nov. 1976), pages 644–654.

[33] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, Vol. 30, No. 2, pages 391–437, 2000. Preliminary version in *23rd STOC*, 1991.

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

[35] C. Dwork and L. Stockmeyer. 2-Round Zero-Knowledge and Proof Auditors. In *34th ACM Symposium on the Theory of Computing*, pages 322–331, 2004.

[36] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating Clique is almost NP-complete. *Journal of the ACM*, Vol. 43, pages 268–292, 1996. Preliminary version in *32nd FOCS*, 1991.

[37] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *11th IEEE Conference on Computational Complexity*, pages 278–287, 1996.

[38] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM Journal on Computing*, Vol. 29 (1), pages 1–28, 1999.

[39] U. Feige and A. Shamir. Witness Indistinguishability and Witness Hiding Protocols. In *22nd ACM Symposium on the Theory of Computing*, pages 416–426, 1990.

[40] U. Feige and A. Shamir. Zero-Knowledge Proofs of Knowledge in Two Rounds. In *Crypto'89*, Springer-Verlag LNCS Vol. 435, pages 526–544, 1990.

[41] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solution to Identification and Signature Problems. In *Crypto86*, Springer-Verlag Lecture Notes in Computer Science (Vol. 263), pages 186–189, 1987.

[42] L. Fortnow, The Complexity of Perfect Zero-Knowledge. In *19th ACM Symposium on the Theory of Computing*, pages 204–209, 1987.

[43] L. Fortnow, J. Rompel and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. 3rd IEEE Symp. on Structure in Complexity Theory*, pages 156–161, 1988.

[44] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On Completeness and Soundness in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 429–442, 1989.

[45] O. Goldreich. A Uniform Complexity Treatment of Encryption and Zero-Knowledge. *Journal of Cryptology*, Vol. 6, No. 1, pages 21–53, 1993.

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

[47] O. Goldreich. *Secure Multi-Party Computation.* Working draft, June 1998.
Available from `http://www.wisdom.weizmann.ac.il/~oded/pp.html`.

[48] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness.* Algorithms and Combinatorics series (Vol. 17), *Springer*, 1998.

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

[50] O. Goldreich. *Foundation of Cryptography – Volume 2.* Working drafts for chapters regarding encryption schemes and signature schemes, 2000. Revised 2002.
Available from `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html`.

[51] O. Goldreich. Concurrent Zero-Knowledge With Timing, Revisited. In *34th ACM Symposium on the Theory of Computing*, pages 332–340, 2002.

[52] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. *Journal of the ACM*, Vol. 33, No. 4, pages 792–807, 1986.

[53] O. Goldreich and J. Håstad. On the Complexity of Interactive Proofs with Bounded Communication. *IPL*, Vol. 67 (4), pages 205–214, 1998.

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

[55] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. *SIAM Journal on Computing*, Vol. 25, No. 1, February 1996, pages 169–192.

[56] O. Goldreich and E. Kushilevitz. A Perfect Zero-Knowledge Proof for a Decision Problem Equivalent to Discrete Logarithm. *Journal of Cryptology*, Vol. 6 (2), pages 97–116, 1993.

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

[58] O. Goldreich, S. Micali and A. Wigderson. How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In *19th ACM Symposium on the Theory of Computing,* pages 218–229, 1987. See details in [47].

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

[60] O. Goldreich, R. Ostrovsky and E. Petrank. Knowledge Complexity and Computational Complexity. *SIAM Journal on Computing*, Vol. 27, 1998, pages 1116–1141.

[61] O. Goldreich and E. Petrank. Quantifying Knowledge Complexity. *Computational Complexity*, Vol. 8, pages 50–98, 1999. Preliminary version in *32nd FOCS*, 1991.

[62] O. Goldreich, A. Sahai, and S. Vadhan. Honest-Verifier Statistical Zero-Knowledge equals general Statistical Zero-Knowledge. In *30th ACM Symposium on the Theory of Computing*, pages 399–408, 1998.

[63] O. Goldreich, A. Sahai, and S. Vadhan. Can Statistical Zero-Knowledge be Made Non-Interactive? or On the Relationship of SZK and NISZK. In *Crypto99*, Springer-Verlag Lecture Notes in Computer Science (Vol. 1666), pages 467–484.

[64] O. Goldreich and S. Vadhan. Comparing Entropies in Statistical Zero-Knowledge with Applications to the Structure of SZK. In *14th IEEE Conference on Computational Complexity*, pages 54–73, 1999.

[65] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, No. 2, pages 270–299, 1984. Preliminary version in *14th STOC*, 1982.

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

[67] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. *Advances in Computing Research: a research annual*, Vol. 5 (Randomness and Computation, S. Micali, ed.), pages 73–90, 1989. Extended abstract in *18th STOC*, pages 59–68, 1986.

[68] J. Håstad, R. Impagliazzo, L.A. Levin and M. Luby. A Pseudorandom Generator from any One-way Function. *SIAM Journal on Computing*, Volume 28, Number 4, pages 1364–1396, 1999.

[69] R. Impagliazzo and M. Yung. Direct Zero-Knowledge Computations. In *Crypto87*, Springer-Verlag Lecture Notes in Computer Science (Vol. 293), pages 40–51, 1987.

[70] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *24th ACM Symposium on the Theory of Computing*, pages 723–732, 1992.

[71] J. Kilian and E. Petrank. An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions. *Journal of Cryptology*, Vol. 11, pages 1–27, 1998.

[72] J. Kilian and E. Petrank Concurrent and Resettable Zero-Knowledge in Poly-logarithmic Rounds In *33rd ACM Symposium on the Theory of Computing*, pages 560–569, 2001.

[73] L.A. Levin. Average Case Complete Problems. *SIAM Jour. of Computing*, Vol. 15, pages 285–286, 1986.

[74] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems. *Journal of the ACM*, Vol. 39, No. 4, pages 859–868, 1992. Preliminary version in *31st FOCS*, 1990.

[75] S. Micali. Computationally Sound Proofs. *SICOMP*, Vol. 30 (4), pages 1253–1298, 2000. Preliminary version in *35th FOCS*, 1994.

[76] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, Vol. 4, pages 151–158, 1991.

[77] T. Okamoto. On relationships between statistical zero-knowledge proofs. In *28th ACM Symposium on the Theory of Computing*, pages 649–658, 1996.

[78] R. Ostrovsky and A. Wigderson. One-Way Functions are essential for Non-Trivial Zero-Knowledge. In *2nd Israel Symp. on Theory of Computing and Systems*, IEEE Comp. Soc. Press, pages 3–17, 1993.

[79] E. Petrank and G. Tardos. On the Knowledge Complexity of NP. In *37th IEEE Symposium on Foundations of Computer Science*, pages 494–503, 1996.

[80] M. Prabhakaran, A. Rosen and A. Sahai. Concurrent Zero-Knowledge Proofs in Logarithmic Number of Rounds. In *43rd IEEE Symposium on Foundations of Computer Science*, 2002.

[81] R. Richardson and J. Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *EuroCrypt99*, Springer LNCS 1592, pages 415–413.

[82] R. Rivest, A. Shamir and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, Vol. 21, Feb. 1978, pages 120–126

[83] A. Sahai and S. Vadhan. A Complete Promise Problem for Statistical Zero-Knowledge. In *38th IEEE Symposium on Foundations of Computer Science*, pages 448–457, 1997.

[84] A. Shamir. IP = PSPACE. *Journal of the ACM*, Vol. 39, No. 4, pages 869–877, 1992. Preliminary version in *31st FOCS*, 1990.

[85] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[86] A.C. Yao. How to Generate and Exchange Secrets. In *27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.