Probabilistic Proof Systems
(an attempt at a popular presentation)

Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot, Israel.

January 19, 2025

Traditional proof systems, ans specifically NP-proof systems, are defined in terms of deterministic verification procedures. However, we can gain a lot if we are willing to take a non-traditional step and allow probabilistic verification procedures. In particular:

- Randomized and interactive verification procedures, giving rise to *interactive proof systems*, seem much more powerful (i.e., more "expressive") than their deterministic counterparts. These probabilistic proof systems demonstrate the advantage of interaction (over one-directional communication).
- Such randomized procedures allow for the introduction of *zero-knowledge proofs*, which are of great theoretical and practical interest. These probabilistic proof systems demonstrate the possibility of decoupling proving from learning (i.e., the possibility of proving an assertion without yielding anything beyond its validity).
- NP-proofs can be efficiently transformed into a (redundant) form (called *probabilistically checkable proofs*) that offers a trade-off between the number of locations examined in the NP-proof and the confidence in its validity.

In all the aforementioned types of probabilistic proof systems, explicit bounds are imposed on the computational complexity of the verification procedure, which in turn is personified by the notion of a verifier. Furthermore, in all these probabilistic proof systems, the verifier is allowed to toss coins and rule by statistical evidence. Thus, all these proof systems carry a probability of error; yet, this probability is explicitly bounded and, furthermore, can be reduced by successive application of the proof system.

On the nature of the claims that will be discussed

Let me start with a couple of orientation remarks. First, the proofs that I am going to discuss are (rigorous) proofs for claims that are self-contained. That is, all information that suffices for the verification of the claim is present (at least implicitly) in the text of the claim, and still verification may be assisted by augmenting the claim with an adequate proof (which refers to the claim's contents). Think of a system of equations of the following form and of the claim that this system

has a non-negative integer solution.

$$\begin{aligned} x+y+z &= 2 \\ x-y &\geq 0 \\ 2y+z &\leq 2 \\ z &\leq 1. \end{aligned}$$

You can verify this claim by yourself, but you must admit that it is easier to verify it when given a proof in the form of a solution to the system (e.g. $x = y = 1$ and $z = 0$). This proof is not explicit in the claim itself, but it is implicit in it. The claim that such a system of equations has no solution is also a claim that is self-contained, but verifying it seems harder (e.g., a proof cannot consist of a single assignment to the variables).

I will use systems of equations with many variables (and many equations) as a running example. This example also illustrates that the claims that I discuss are mundane ones (rather than being fundamental theorems such as the Pythagorean theorem or Fermat's Last Theorem).

Let me comment that the foregoing example can be used to illustrated the three types of proof systems we shall consider next. A traditional (NP-proof) of the claim that a specific system of equations has a solution is provided by the solution itself. But there seems to be no NP-proof for the claim that such a specific system has no solution, and yet interactive (randomized) proofs can do the job. Furthermore, it is not clear how to prove that such a specific system has a solution without yielding any additional information about such a solution, and yet zero-knowledge proofs are defined as doing that.

An apology: Don't ask me what does NP stand for. It is a bad acronym, which is typical of the poor public relations skills of the theory of computation. Consequently, the fascinating intellectual contents of the theory of computation is rarely communicated to the non-specialists, let alone to the general public.[1] This article is a modest attempt at correcting this sour state of affairs.

Advanced comment: I insisted on integer solutions (to systems of equation) because in that case even the solvability of systems of linear equations is NP-complete. Hence, whatever is shown about the (integer) solvability of a system of (linear) equations also holds for any other class of claims in NP (equiv., claims having NP-proof systems).

Traditional Proofs: NP-Proof System

The glory attributed to the creativity involved in finding proofs makes us forget that it is the less glorified procedure of verification that gives proofs their value. Conceptually speaking, proofs are secondary to the verification procedure; whereas technically speaking, proof systems are defined in terms of their verification procedures.

[1] The fact that any living adult is aware of the revolutionary impact of the computing technology on our society, does not contradict the fact that only few of them are familiar with the theory of computation. This contrast is not so surprising, because people seem so overwhelmed by the wonders of this technology that they do not get to wonder about the theory underlying it. Furthermore, people tend to think of computation in the concrete terms in which they have lastly encountered it rather than in general terms.

The notion of a verification procedure assumes the notion of computation and furthermore the notion of efficient computation. This implicit stipulation is made explicit in the definition of NP-proof systems, where efficient computation is associated with deterministic algorithms whose running-time increases moderately with the length of their inputs. (The standard convention is to model such moderately increasing functions by polynomials.) Hence, NP-proof systems employ efficient verification procedures that examine proofs that are of moderate length (in terms of the length of the claim being proved).

A Detour: The P-vs-NP question.

Let me take a detour and challenge our intuition that proofs (i.e., NP-proofs) are of value. This intuition is based on the belief that it is easier to verify a proof for given claim than to evaluate the correctness of the claim by ourselves. Think about the running example of a system of equations and the claim that the system has an integer solution, but think of the case that the system has 100 equations in 100 variables. It seems infeasible to check whether such a system has an integer solution, but it is relatively easy to verify the correctness of a solution once it is given to us. Hence, such a solution is a proof (i.e., an NP-proof) to the validity of the claim (that the system has a solution). *But is it a fact that it is infeasible to check whether such a system has an integer solution* (when not given a proof)*?*

The answer is that we don't know. This is the famous P-vs-NP question. Indeed, in one formulation, the P-vs-NP question asks whether proofs are of any value? That is, *is it fundamentally easier to verify claims when given adequate proofs than it is to check the validity of these claims without any additional information* (i.e., a proof)? The common belief is that the answer is positive, and the very notion of a "proof" would become meaningless otherwise. The correctness of this common belief is arguably the most important open problem of computer science.

Back from the detour: A close look at NP-proof system

Recall that the foregoing exposition of NP-proof systems referred to a single (efficient) algorithm – the verification procedure. Let me spell out the requirement from the verification procedure, which are called *completeness* and *soundness*. These requirements refer to a class of claims that is supposedly handled by the proof system (e.g., claims about the existence of a non-negative integer solution to a system of equations).

***Completeness*:** This condition requires that valid claims (in the class) have proofs that are accepted (as correct) by the verification procedure.

***Soundness*:** This condition requires that invalid claims (in the class) do not have alleged proofs that are accepted (as if they were correct) by the verification procedure.

These two conditions are very intuitive and any notion of a proof system should postulate them: They assert that valid claims can be proved in the system, whereas invalid claims cannot be proved in it. Consider the example of claims that refer to systems of equations. In that case, a solution to the system of equations constitutes an NP-proof for the claim that the system has a solution, where the verification procedure consists of substituting the variables by the provided values and evaluating the corresponding arithmetic expressions.

I find it useful, especially in the context of what follows, to present proof systems as two-party games between a verifier, who is using the verification procedure, and a potential prover, who provide the latter procedure with alleged proofs. Indeed, a proof system refers explicitly to the verification task, which is personified by the verifier, but it also refers implicitly to a prover as a personification of the source of potential proofs.

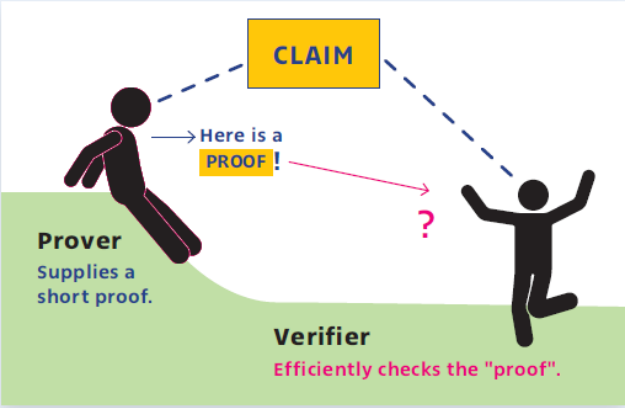

Figure 1: Traditional Proofs: Unidirectional Communication

With this personification in place, we can think of NP-proofs as a unidirectional communication from the prover to the verifier: The prover writes and sends an alleged proof, and the verifier reads and verifies it (see Figure 1). Indeed, the terms "writes" and "reads" correspond to the traditional view of proofs as being texts, written in books and articles, that are read by learners.

The foregoing personification also calls for a personification of the completeness and soundness conditions: Completeness means that the verifier can be convinced of the correctness of valid claim by adequate proofs, which may be provided by a (potentially more knowledgeable) prover, whereas soundness means that nobody can fool the verifier into accepting a false proof to an invalid claim. In other words, *completeness* means that there is a strategy (for the prover) to convince the verifier of valid claims, whereas *soundness* means that nobody can convince the verifier of invalid claims.

Beyond NP-proof system

A key feature of NP-proofs, which is taken for granted, is that they employ (efficient) deterministic verification procedures (i.e., computational process in qhich each step is uniquely determined by the current sitaution). I did mention taht the verification procedure is deterministic in the foregoing discussion, but I bet that it went unnoticed, because it is assumed by default when discussing proofs. (One reason for this assumption is that probabilistic verification means a probability of error, which seems contradictory to the notion of a proof; but I disagree with the latter view.)

However, as hinted in the title of this article, I am going to deviate from this convention and focus on (efficient) probabilistic verification procedures, which yield fascinating types of *probabilistic proof systems*. A common feature of these proof systems is that they carry a probability of error, but this probability is explicitly bounded and can be reduced by repetitions.

Interactive (and Randomized) Proofs

Viewing proofs as two-party games between verifiers and potential provers is instrumental for the introduction of interactive proof systems. Given such a view, it is natural to consider bidirectional communication (i.e., a full-fledged interaction between the prover and the verifier) rather than confine ourselves to unidirectional communication (i.e., a single message, constituting an alleged proof, that the prover sends to the verifier). This leads to the notion of *interactive proofs* (see Figure 2); indeed, interactive proofs are dynamic processes rather than static objects.

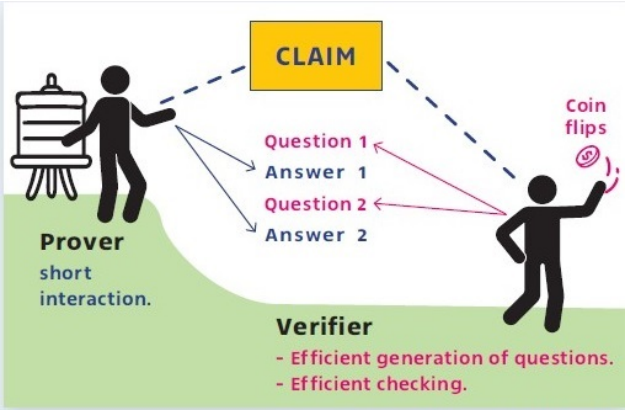

Figure 2: Interactive Proofs: Bidirectional Communication

Although the word "randomness" does not appear in the term "interactive proofs", one can easily see that randomness is essential for the advantage of interactive proofs over NP-proofs. In contrast, if the verifier is deterministic, then the prover can predict all the verifier's messages. In such a case, the prover can just send an NP-proof that records the interaction between the two original parties. The lesson here is that it makes no sense to interact with a party that is easily predictable; in contrast, interaction with another party may be beneficial only if the other party is either unpredictable (i.e., randomized) or has more computational resources (and so can do work for you).

But what about the fundamental notions of completeness and soundness? When using randomized (and interactive) verification strategies, these notions have to be refined so to reflect a meaningful notion of statistical evidence. Specifically, in the context of interactive proof systems, completeness and soundness are defined as follows (with respect to a fixed strategy employed by the verifier).

***Completeness*:** If the claim is valid, then there is a strategy for the prover to make the verifier accept the claim with probability 1.

(Requiring that the verifier accepts with probability at least 2/3 is also meaningful. Furthermore, an interactive proof with such a "completeness error" can be converted into one that has no completeness error.)

***Soundness*:** If the claim is invalid, then any strategy employed by the prover would fail to fool the verifier with probability at least 1/2.

Getting fooled with probability 1/2 sounds alarming, but this bounded probability (of being fooled) can be reduced by repeated invocations of the proof system.

As in the case of NP-proofs, completeness postulates that valid claims can be proved in the system, and soundness postulates that invalid claims can not be proved in it (except for with bounded probability, which can be reduced by repetitions).

An illustration

Suppose that you are given two systems of equations, as in the running example, and you wonder whether they are "isomorphic" in the sense that changing the variable names and the order of equations in the first system yields exactly the second system. An NP-proof that specifies the variable-substitution and re-ordering can easily convince you that the two systems are isomorphic. But what about getting a proof that the systems are not isomorphic?

We do not know of an NP-proof system for non-isomorphism, but here interactive proofs come to the rescue. What you as the verifier can do is select at random one of the two systems of equations, randomly replace its variable names and reorder its equations, and send the result to the prover asking it to tell you which of the two systems you started from. Note that soundness follows from the fact that if the systems of equations are isomorphic, then there is no way for the prover to answer correctly with probability exceeding 1/2. As for completeness, it follows from the fact that if the systems of equations are non-isomorphic, then so are also their random renaming and reordering.

A general result

Continuing with the running example, consider the more natural problem of checking whether a system of equations has no integer solution. As noted before, NP-proofs can be used to show that such systems do have an integer solution, but it is widely believed that NP-proofs cannot be used to show that such systems have no integer solution. Again, interactive proof systems come to the rescue, but they are too complex to present here. Let me just restate the result more clearly: *There exists interactive proofs for showing that a system of equations has no integer solution.*

More generally, consider claims that assert the existence of something that is implicit in the data that is part of the claim, and assume that once presented this thing can be easily verified as valid. In our running example, this thing is an integer solution to the system of equations, which is viewed as the data. Evidently, NP-proof can be utilized for proving the existence of a thing of interest in the data. The amazing fact is that, in this case, interactive proofs can be utilized for proving the non-existence of that thing in the data.

In other words, whenever there exists an NP-proof system for a class of claims (e.g., solvability of a system of equation), there exists an interactive proof system for the class of opposite claims (e.g., non-solvability of a system of equation). (Of course, on any specific instance, only one of the proof systems can convince the corresponding verifier; the point is that proof systems exist for each of the two opposite types of claims.)

Two comments

In the interactive proof used for the non-isomorphism of two systems of equations, it was essential that the verifier's random choices were kept secret from the prover, who only received their effect

on the common input. It turns out that this use of secret randomness is inessential in the sense that there exists an alternative proof system that does not use secret randomness (alas is more complex). In general, every class of claims that has an interactive proof system also has one in which the verifier sends each random choice it makes; in other words, with respect to the mere existence of interactive proof systems, random queries are as powerful as sophisticated ones.

Throughout the foregoing discussion, our focus was on the complexity of the verification task, and we totally ignored the complexity of the proving task. A more refined study considers the complexity of both tasks, and focuses on interactive proof systems in which proving is not much harder than deciding (without a proof) and verification is significantly easier (than deciding). Such interactive proof systems, which are of evident practical relevance, are called *doubly-efficient*.

Zero-Knowledge Proofs

Typically, proof yield much more than the mere validity of the claim being proved. In fact, learning proofs is often considered the best way of understanding the actual contents and context of the claim. In contrast, zero-knowledge proofs are designed to yield nothing beyond the validity of the claim being proved. This feature is indeed confusing and counter-intuitive. Actually, it cannot exist in the context of NP-proof systems; hence, it is formulated in the context of interactive proof systems and is a feature of some of them.

Consider, for example, the interactive proof for non-isomrphism (of two systems of equations). The verifier in that interactive proof learns nothing beyond the validity of the claim, because the prover's answer is merely the initial choice made by the verifier. Recall that the crux of this proof system is that the prover can correctly determine the initial choice made by the verifier in the case the two systems are not isomorphic (although it cannot have any advantage over guessing if the two systems are isomorphic).

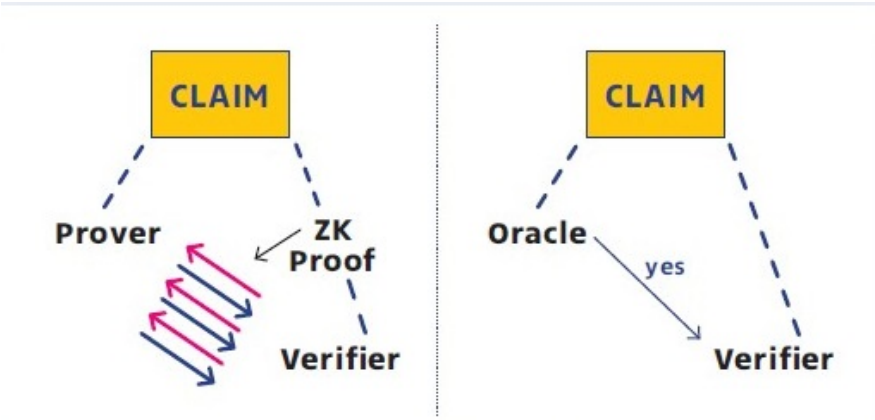

Figure 3: Zero-Knowledge Proofs: The Real World versus the Ideal World

A good question at this point is how to formalized the notion of "learning nothing beyond the validity of the claim being proved". Loosely speaking, this notion is captured by saying that *whatever the verifier can efficiently compute after getting a zero-knowledge proof can be efficiently computed based on the claim itself* (assuming it is valid). That is, the actual definition compares two settings – the real world in which the zero-knowledge interactive proof takes place and an ideal

world in which the verifier is assured by a trusted oracle that the claim is valid (see Figure 3). The definition postulates that whatever the verifier can compute after the interaction in the real world, can be computed as easily in the ideal world.

Hence, the only reason for the verifier to engage in a zero-knowledge interactive proof is gaining assurance that the claim is valid, which is something he would have gained in an ideal world (which does not exist). Needless to say, this fascinating feature is very useful in cryptography, where parties may want to prove that they are behaving correctly (in accordance with their secrets) without yielding their secrets.

Consider our running example of a system of equations. In this case, a zero-knowledge proof that the system has an integer solution does not yield any partial information about any such solution. It merely convinces the verifier that such a solution exists. Amazingly enough, such a zero-knowledge proof exists, assuming that one-way functions exist, which is a necessary assumption for most of modern cryptography (and, in particular, for the existence of secure encryption schemes and unforgeable digital signatures).

In general, whenever there exists an NP-proof system for a class of claims (e.g., solvability of a system of equation), there exists a zero-knowledge interactive proof system for this class. Furthermore, the prover strategy in this zero-knowledge proof is efficient, provided that it is given an NP-proof for the claim to be proved. (We comment that any class of claims that has an interactive proof stystem also has a zero-knowledge one.)

Probabilistically Checkable (Written) Proofs

Let us now revisit the traditional NP-proof systems (i.e., forget about interactive proof systems), but keep randomization at our disposal.

Typically, the verification of written proofs (i.e., NP-proofs) requires reading them entirely. If you skip some parts, which you don't know from elsewhere, then you run the risk of being fooled by a false argument. In contrast, probabilistically checkable proofs (hereafter referred to as PCPs) are designed to allow for a statistical evaluation of their validity based on reading tiny parts of them. As will be illustrated below, these proofs must be in redundant form and the choice of parts to read should be randomized (see Figure 4).

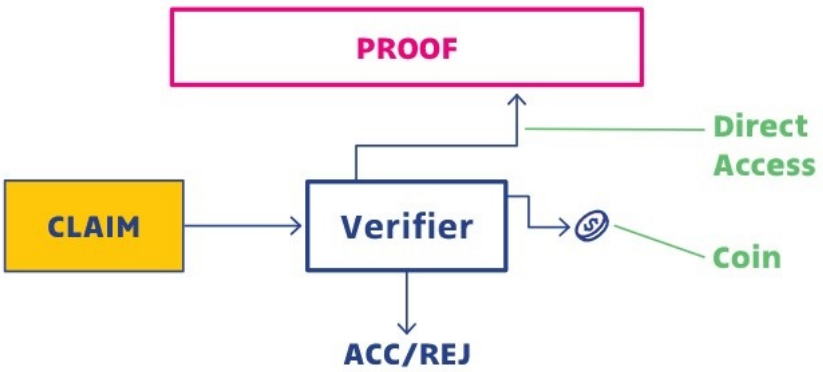

Figure 4: PCPs: The verifier has direct access to bits of the proof

Consider, for example, the claim that a system of equations has an integer solution and the NP-proof that consists of one the solutions. This NP-proof cannot possibly be a PCP, because the system of equations may be unsolvable due to a contradiction regarding the value imposed on a single variable. This suggests that the desired PCP must have redundancy. Regarding the necessity of randomness, note that deterministic verification based on few locations means that the NP-proof is effectively very short, which contradicts common beliefs regarding the class NP.

The ideas underlying the construction of PCPs are too complex for me to attempt an illustration here. Instead, let me only state the fact that *any NP-proof can be efficiently converted into a redundant form such that reading three bits of it offers meaningful completeness and soundness properties*. Specifically, valid NP-proofs of valid claims are converted to PCPs that are accepted with probability 1, whereas alleged (false) proofs for invalid claims are rejected with probability at least 0.49 (i.e., any constant smaller than 1/2).

Let me mention that reading three bits in a proof is the very minimum that offers meaningful completeness and soundness properties, and that a rejection probability bound of 0.49 (standing for any constant smaller than 1/2) is also optimal (with respect to reading three bits).

The construction of PCP systems had an enormous impact of the study of approximation problems. This is the case because the evaluation of the acceptance probability of a fixed verifier on a given claim is an optimization problem, and it can be reduced to many natural optimization problems. (Understanding the last assertion may require familiarity with the theory of NP-completeness, in which the notion of a reduction plays a pivotal role.)

Credits and Bibliography

I have intentionally avoided mentioning the names of the researchers who deserve most credit for the theory briefly reviewed in this exposition. Such credits, which are a must in scholarly articles, add little to the general public. Furthermore, mentioning few names creates an artificial gap between "celebrities" who are mentioned and "common researchers" who are ignored. This stands in contrast to the reality in which science is a communal project.

I am unaware of good popular expositions of the theory of computation, let alone of probabilistic proof systems. The following books and surveys are aimed at undergraduate students in computer science and mathematics. In particular, NP-proofs are at the core of [5] as well as of [4, Chap. 2], whereas probabilistic proof systems are the subject of [3] and [4, Chap. 9]. The briefly-mentioned notion of doubly-efficient interactive proof systems is surveyed in [7]. The cryptographic applications of zero-knowledge proofs are at the core of [1, Chap. 4] and [2, Chap. 7]. The study of PCPs led and is related to the study of property testing at large, which is the subject of [6].

References

[1] Oded Goldreich. Foundation of Cryptography: Volume 1 – Basic Tools. Cambridge University Press, 2001.

[2] Oded Goldreich. Foundation of Cryptography: Volume 2 – Basic Applications. Cambridge University Press, 2004.

[3] Oded Goldreich. Probabilistic Proof Systems – A Primer. In *Foundations and Trends in Theoretical Computer Science*, Volume 3, Issue 1, 2007.

[4] Oded Goldreich. Computational Complexity: A Conceptual Perspective. Cambridge University Press, 2008.

[5] Oded Goldreich. P, NP, and NP-Completeness: The Basics of Complexity Theory. Cambridge University Press, 2010.

[6] Oded Goldreich. Introduction to Property Testing. Cambridge University Press, 2017.

[7] Oded Goldreich. On Doubly-Efficient Interactive Proof Systems. In *Foundations and Trends in Theoretical Computer Science*, Volume 13, Issue 3, 2018.

Figures credit: Courtesy of the Israel Academy of Sciences and Humanities.