# On some non-cryptographic works of Goldwasser and Micali*

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

May 5, 2019

**Abstract**

While this book focuses on the contributions of Goldwasser and Micali to Cryptography, their contributions to other areas of computer science are immense too. The current chapter endeavors to briefly review some of these works.

## Contents



*To appear as Chapter 19 in an ACM book celebrating the work of Goldwasser and Micali.

While this book focuses on the contributions of Goldwasser and Micali to Cryptography, their contributions to other areas of computer science are immense too. In particular, while the original works reproduced in this book were all motivated by cryptographic considerations and made significant contributions to the foundations of cryptopgraphy, all of them have had a tremendous influence also outside of cryptography. In fact, another chapter of this book traces the influences that these works have had on complexity theory, but the story does not end there.

A different part of the story evolves around works of Goldwasser and Micali that are not naturally classified as belonging to cryptography. The current chapter endeavors to briefly review some of these works.[1] For each of the selected works, we shall reproduce the original abstract, and make a few additional comments about the work.

## 1 An $O(\sqrt{|V|}\cdot|E|)$-time Algorithm for Finding Maximum Matching in General Graphs

The work of Micali and Vazirani [40] still holds the record for the fastest algorithm known for finding a maximum matching in general graphs, which is one of the most classical problems in graph algorithms [19]. (For a brief historical account of the problem, the interested reader is referred to [40].) The time bound of this algorithm (i.e., $O(\sqrt{|V|}\cdot|E|)$), matches the bound for the bipartite case [35], which is considerablly simpler. The source of difficulty is the complex "blossom structure" introduced by Edmonds [18]. The abstract of the conference version of [40] reads as follows.

> In this paper we present an $O(\sqrt{|V|}\cdot|E|)$ algorithm for finding a maximum matching in general graphs. This algorithm works in 'phases'. In each phase a maximal set of disjoint minimum length augmenting paths is found, and the existing matching is increased along these paths.
>
> Our contribution consists in devising a special way of handling blossoms, which enables an $O(|E|)$ implementation of a phase. In each phase, the algorithm grows Breadth First Search trees at all unmatched vertices. When it detects the presence of a blossom, it does not 'shrink' the blossom immediately. Instead, it delays the shrinking in such a way that the first augmenting path found is of minimum length. Furthermore, it achieves the effect of shrinking a blossom by a special labeling procedure which enables it to find an augmenting path through a blossom quickly.

While the original publication [40] provided a detailed description of the algorithm, it did not provide its analysis, and the authors' intentions of publishing a full analysis at a later stage were never materialized. A full analysis, which is based on new graph-theoretic structural facts and a revised definition of blossoms, has been provided by Vazirani [45]. Alternative algorithms meeting the same time bound as [40] have appeared subsequently to it (see, e.g., [23]).

[1] The works of Goldreich and Goldwasser [27] and Goldwasser, Kalai, and Rothblum [31] were omitted from our selection since they will be covered by other surveys in this book (see Chapters 21 and 24, respectively).

## 2 Certifying Almost All Primes using Elliptic Curves

The work of Goldwasser and Kilian [32] predated the deteministic primality testers of [2] by almost two decades. As the following abstract states, at the time, primality testing were either randomized or relied on unproven conjectures. The randomized tests place the set of primes in co$\mathcal{RP}$; that is, they always rule that a prime is a prime, but they may rule with small probability that a composite number is a prime. The randomized procedure provided by [32] efficiently generates (efficiently and deterministically verifiable) certificates of primality, which always vouches that a prime number is indeed a prime, for almost all primes. Indeed, on some primes, the procedure may always fail to produce a certificate, but it never generates false "certificates" for composite numbers. In some sense, this work asserts that the set of primes is in "average-case $\mathcal{RP}$" (or "typical $\mathcal{RP}$"). The abstract of the conference version of [32] reads as follows.

> This paper presents a new probabilistic primality test. Upon termination the test outputs "composite" or "prime", alone with a short proof of correctness, which can be verified in deterministic polynomial time. The test is different from the tests of Miller [M], Solovay-Strassen [SS], and Rabin [R] in that its assertions of primality are certain, rather than being correct with high probability or dependent on an unproven assumption.
>
> The test terminates in expected polynomial time on all but at most an exponentially vanishing fraction of the inputs of length $k$, for every $k$. This result implies:
>
> - There exist an infinite set of primes which can be recognized in expected polynomial time.
> - Large certified primes can be generated in expected polynomial time.
>
> Under a very plausible condition on the distribution of primes in "small" intervals, the proposed algorithm can be shown to run in expected polynomial time on every input. This condition is implied by Cramers conjecture.
>
> The methods employed are from the theory of elliptic cures over finite fields.

The starting point of this work is Pratt's demonstration [42] that the set of primes is in $\mathcal{NP}$; that is, the fact that there exist (efficiently verifiable) certificates of primality, albeit these certificates may not be easy to find. This is the case, because these certificates are defined recursively such that the certificate for a prime $P$ consists of a generator $G$ of $\mathbb{Z}_P^*$ (i.e., a primitive element modulo $P$), the prime factorization of $P-1$, and certificates for primality for each of its prime factors. The prime factorization is used to verify that $G$ has (multiplicative) order $P-1$ (in $\mathbb{Z}_P^*$), which in turn implies that $P$ must be a prime.

Specifically, a valid certificate has the form $((P_1, e_1, C_1), ..., (P_t, e_t, C_t), G)$ such that $P - 1 = \prod_{i=1}^{t} P_i^{e_i}$, the order of $G$ in $\mathbb{Z}_P^*$ is $P-1$ (i.e., $G^{P-1} \equiv 1 \pmod{P}$ but $G^{(P-1)/P_i} \not\equiv 1 \pmod{P}$ for each $i$), and $C_i$ is a certificante for primality of $P_i$. The validity of this certificate relies on the fact that $G$ may have order $P-1$ in $\mathbb{Z}_P^*$ if and only if $P$ is a prime. More abstractly, primes $P$ yield groups of predetermined order, denoted $\mathtt{ord}(P)$, whereas composite numbers yield groups of a different order (i.e., if $P$ is composite, then $|\mathtt{Z}_P| \neq \mathtt{ord}(P) = P-1$). The problem with generating such certificates is that it calls for factoring $P-1$, which seems hard.

Suppose, instead, that given a prime $P$ and random choices $\omega$, we can define a group $R_{P,\omega}$, of order $\texttt{ord}(P,\omega) = P \pm o(P)$ such that the function $\texttt{ord}$ and the group operation are easy to compute. If we can efficiently generate (possibly at random) an element of order $\texttt{ord}(P,\omega)$ in that group, and if for composite $P$ the "order" of the "structure" $R_{P,\omega}$ disagrees with $\texttt{ord}(P,\omega)$, then the foregoing reasoning would apply here too. The benefit is that, now, generating a certificate for $P$ calls for factoring $\texttt{ord}(P,\omega)$ rather than factoring $P-1$, and if $\texttt{ord}(P,\cdot)$ is random enough then we are in business. Specifically, if $\texttt{ord}(P,\cdot)$ is uniformly distributed in a sufficiently large interval around $P$, then we can factor $\texttt{ord}(P,\cdot)$ often enough, since in such a case with probability at least $\Omega(1/\log P)$, it holds that $\texttt{ord}(P,\cdot) = 2Q$ for a prime $Q$. This is essentially what happens when using (suitably) random Elliptic Curves mod $P$, and the complication arise because the relevant interval has size $\sqrt{P}$ (rather than, say $P/\mathrm{poly}(\log P)$).

Hence, the reviewed work asserted that the set of primes is in "average-case $\mathcal{RP}$" (or "typical $\mathcal{RP}$"), and this begged the challenge of showing that the set of primes is actually in $\mathcal{RP}$. The challenge was met by Adleman and Huang [1]. Fifteen years later, Agrawal, Kayal, and Saxena [2] showed that the set of primes is actually in $\mathcal{P}$.

## 3 Private Coins versus Public Coins in Interactive Proof Systems

The work of Goldwasser and Sipser [34] predated the discovery of the vast power of interactive proof systems, and, in particular, the $\mathcal{IP} = \mathcal{PSPACE}$ Theorem [38, 44]. The starting point of [34] is the fact that Babai [5] defined Arthur-Merlin games as a restricted form of interactive proof systems, which were defined before by Goldwasser, Micali, and Rackoff [33], where the restriction is that the verifier is only allowed to make uniformly selected queries (a.k.a use public coins). This difference is not surprising given that Goldwasser, Micali, and Rackoff sought to capture the most general notion of a proof system (with efficient verification) [33], whereas Babai sought a minimal extension of the class $\mathcal{NP}$ (in order to place some specific computational problem in it) [5]. Surprisingly, Goldwasser and Sipser [34] showed that the aforementioned restriction does not weaken the expressive power of the system; put differently, asking random questions is as good as asking cleverly selected questions (i.e., questions that are the result of an arbitrary probabilistic polynomial-time computation, whose coins are not revealed to the prover but may be re-used when examining the prover's answers). The abstract of the conference version of [32] reads as follows.

> An interactive proof system is a method by which one party of unlimited resources, called the *prover*, can convince a party of limited resources, called the *verifier*, of the truth of a proposition. The verifier may toss coins, ask repeated questions of the prover, and run efficient tests upon the provers responses before deciding whether to be convinced. This extends the familiar proof system implicit in the notion of NP in that there the verifier may not toss coins or speak, but only listen and verify. Interactive proof systems may not yield proof in the strict mathematical sense: the "proofs" are probabilistic with an exponentially small, though non-zero chance of error.
>
> We consider two notions of interactive proof systems. One, defined by Goldwasser, Micali and Rackoff [GMR] permits the verifier a coin that can be tossed in *private*, i.e., a secret source of randomness. The second, due to Babai, [B] requires that the outcome of the verifiers coin tosses be *public* and thus accessible to the prover.

> Our main result is that these two systems are equivalent in power with respect to language recognition.
>
> The notion of interactive proof system may be seen to yield a probabilistic analog to NP much as BPP is the probabilistic analog to P. We define the *probabilistic, nondeterministic, polynomial time Turing machine* and show that it is also equivalent in power to these systems.

We stress that the result actually shown is stronger: The authors showed that any $r$-round interactive proof system can be emulated by an $(r+3)$-round interactive proof system of the public-coin type. We comment that the mere fact that interactive proof system can be emulated by interactive proof system of the public-coin type follows from the subsequent demonstration that $\mathcal{IP} = \mathcal{PSPACE}$, because the original demonstration actually shows that any set in $\mathcal{PSPACE}$ has a *public-coin* interactive proof system [38, 44] (whereas $\mathcal{IP} \subseteq \mathcal{PSPACE}$, where $\mathcal{IP}$ denotes the class of sets having (general) interactive proof systems).

The fact that private coins are of no real help came as a surprise, especially in light of the interactive proof system presented around the same time for Graph Non-Isomorphism, since that proof system makes essential use of private coins [29]. In that proof system, the verifier selects at random one of the two graphs, sends a randomly permuted (or relabeled) version of it to the prover, and accepts if and only if the prover identifies correctly which graph was chosen. In this specific case, the public-coin proof system derived by [34] amounts to proving a lower bound on the size of automorphism group of the graph consisting of both graphs (and an upper bound on the size of of automorphism groups of each of the individual graphs).[2]

In general, a key ingredient of the construction of [34], is a public-coin protocol, known as the *lower bound protocol*, that allows one party to prove to another that the size of a set exceeds some given number (provided that the set is in $\mathcal{NP}$).[3] This protocol, which is closely related to a "random selection" protocol, was used extensively in subsequent works.

# 4 An Optimal Randomized Protocol for Synchronous Byzantine Agreement

The work of Feldman and Micali [21] presents a constant-round randomized Byzantine Agreement protocol for a synchronous communication model with private channels. As in [10], the private-channel model allows to abstract away intractability assumptions and cryptographic tools, although implementing this clean model on a network of insecure channels does require such assumptions and tools. The protocol improved over a prior protocol of Bracha [15] that used logarithmically many rounds (and intractability assumptions). The conference version of [21] had no abstract, and the abstract of the journal version reads as follows.

> Broadcasting guarantees the recipient of a message that everyone else has received the same message. This guarantee no longer exists in a setting in which all communication is person-to-person and some of the people involved are untrustworthy: though he may

[2] An upper bound on the size of automorphism group of a graph $G$ follows by a lower bound on the number of different graphs that are obtained by relabeling the vertices of $G$.

[3] In the general case, when claiming a lower bound of $N$, the prover is confined to an $1/N$ fraction of the original set. Hence, if the set is smaller than $N$, then the prover may be confined to an empty subset of it.

claim to send the same message to everyone, an untrustworthy sender may send different messages to different people. In such a setting, Byzantine agreement offers the "best alternative" to broadcasting. Thus far, however, reaching Byzantine agreement has required either many rounds of communication (i.e., messages had to be sent back and forth a number of times that grew with the size of the network) or the help of some external trusted party.

In this paper, for the standard communication model of synchronous networks in which each pair of processors is connected by a private communication line, we exhibit a protocol that, in probabilistic polynomial time and without relying on any external trusted party, reaches Byzantine agreement in an expected constant number of rounds and in the worst natural fault model. In fact, our protocol successfully tolerates that up to 1/3 of the processors in the network may deviate from their prescribed instructions in an arbitrary way, cooperate with each other, and perform arbitrarily long computations.

Our protocol effectively demonstrates the power of randomization and zero-knowledge computation against errors. Indeed, it proves that "privacy" (a fundamental ingredient of one of our primitives), even when is not a desired goal in itself (as for the Byzantine agreement problem), can be a crucial tool for achieving correctness.

Our protocol also introduces three new primitives – graded broadcast, graded verifiable secret sharing, and oblivious common coin – that are of independent interest and may be effectively used in more practical protocols than ours.

Byzantine Agreement, introduced by Pease, Shostak, and Lamport [41], is considered the archetypical problem of processor coordination, which is a central theme in Distributed Computing [37]. Here, we consider randomized protocols for Byzantine Agreement in the synchronous model, since those bypass the linear (in the number of parties) lower bounds on the round complexity of deterministic protocols in this model.[4] The protocol of Feldman and Micali [21] runs for a constant number of rounds and satisfies the following conditions: (1) in *each* possible execution, each of the parties either terminates with the same value $v$ or terminates with failure, and if all honest parties enter with the same value, then $v$ equals this value; and (2) with constant probability, over all possible executions, no party terminates with failure.

We comment that the private channels used by [21] are essential for a constant-round randomized Byzantine Agreement protocol in the full-fledged malicious model considered by [21]: In fact, even in weaker (adaptive) models with no private channels, a number of rounds that grows roughly as the square root of the number of parties is necessary [8]. On the other hand, the full-fledged without private channels does allow for randomized Byzantine Agreement protocols with a sublinear number of rounds [17].[5]

[4] In the asynchronous model, deterministic protocols face an impossibility result, whereas randomized protocols do exist. But our focus here is on the synchronous model.

[5] The models considered in [17, 21, 8] are adaptive in the sense that an external adversary may adaptively select parties to corrupt during the execution of the protocol (and control their actions). In contrast, in non-adaptive models, the faulty parties are determine (arbitrarily) before the execution starts. A randomized Byzantine Agreement protocols with a logarithmic number of rounds was later shown in the non-adaptive malicious model with no private channels [11].

## 5 PCPs and the Hardness of Approximating Cliques

The work of Feige, Goldwasser, Lovász, Safra, and Szegedy [20] pioneered the study of (what become later known as) "probabilistically checkable proofs" and its relation to the study of approximation problems. A probabilistically checkable proof system for a set $S$ is defined via a probabilistic polynomial-time oracle machine, called a *verifier*, that satisfies the following *completeness* and *soundness* conditions: For every $x \in S$ there exists a proof $\pi$ such that $\Pr[V^\pi(x)=1] = 1$, whereas for every $x \notin S$ and every $\pi$ it holds that $\Pr[V^\pi(x) = 1] \leq 1/2$. For functions $r, q : \mathtt{N} \to \mathtt{N}$, we let $\mathcal{PCP}[r, q]$ denote the class of sets that have a (non-adaptive) probabilistically checkable proof system of randomness complexity $r$ and query complexity $q$. The reviewed work [20] shows that $\mathcal{NP} \subseteq \mathcal{PCP}[\widetilde{O}(\log), \widetilde{O}(\log)]$, which is a "scale down" of a prior result [6] asserting that $\mathcal{NEXP} = \mathcal{PCP}[\mathrm{poly}, \mathrm{poly}]$. Feige, Goldwasser, Lovász, Safra, and Szegedy [20] also showed that deciding sets in $\mathcal{PCP}[r, q]$ is reducible in $\mathrm{poly}(2^{t\cdot(r+q)})$-time to approximating the largest clique in a $2^{t\cdot(r+q)}$-vertex graph up to a factor of $2^t$. The abstract of the conference version of [20] reads as follows.

> We consider the computational complexity of approximating $\omega(G)$, the size of the largest clique in a graph $G$. We show that
>
> 1. If there is an approximation algorithm in $\mathbf{P}$ for $\omega(G)$ within some constant factor, then $\mathbf{NP} \subseteq DTIME(n^{O(\log\log n)})$.
> 2. If there is an approximation algorithm in $\widetilde{\mathbf{P}}$ $(= \cup_{k>0} DTIME(n^{\log^k n}))$ for $\omega(G)$ within a factor of $2^{\log^{1-\epsilon} n}$ (for some $\epsilon > 0$), then $\mathbf{NP} \subseteq \widetilde{\mathbf{P}}$.
>
> We conclude that if such approximation procedures exist, then EXPTIME = NEXPTIME and $\mathbf{N}\widetilde{\mathbf{P}} = \widetilde{\mathbf{P}}$.
>
> This work uses the theorem of Babai, Fortnow and Lund that NEXPTIME has multi-prover interactive proofs. For our purpose, we scale down [BFL90]'s protocol to the NP level, and improve its efficiency. Of independent interest is our simpler proof of correctness for the multi-linearity test.

We mention that independently of [20], Babai, Fortnow, Levin, and Szegedy [7] showed that $\mathcal{NP} = \mathcal{PCP}[O(\log), \mathrm{poly}(\log)]$. Their results were stated in terms of what became later known as PCPs for promiximity (cf., e.g., [12]); specifically, they showed a PCP for proximity for NP-complete sets (which encode standard NP-sets) in which the verifier runs in polylogarithmic time.

Subsequent work of Arora, Lund, Motwani, Safra, Sudan and Szegedy [4, 3] resulted in the celebrated PCP Theorem asserting that $\mathcal{NP} = \mathcal{PCP}[O(\log), O(1)]$. A vast amount of research followed. Most of it has been directed towards extending and utilizing the *PCP-to-inapproximabilty connection*, often while optimizing some parameter of the PCP system that governs the quality of the said connection. This type of research is the focus of Chapter 22. In addition, much research has been devoted to exploring various aspects of the PCP Theorem and providing various versions of it, while envisioning these systems as being actually applied to verify the correctness of computations. In such settings, the proof length seems a dominant parameter (and the interested reader is referred to [26, Chap. 13]).

We conclude this review with two comments. First, we note that employing the PCP-to-inapproximabilty connection may call for optimizing parameters significantly differently than when

seeking to apply the PCP system for actual verification. For example, the PCP-to-clique connection used in [20] motivated the authors of [20] to minimize the value of $r + q$ (using the setting $r(n) = q(n) = \widetilde{O}(\log n)$), whereas the application to actual verification motivated the authors of [7] to minimize $r$ first and only then minimize $q$ (using the setting $r(n) = (1+\epsilon) \cdot \log n$ for arbitrary small constant $\epsilon > 0$, and $q(n) = \mathrm{poly}(\log n)$).[6] Second, we mention that [7, 20] used the formulation of probabilistically checkable proofs, which was shown by Fortnow, Rompel, and Sipser [22] to be equivalent to the formulation of multi-prover interactive proofs, which in turn was introduced by Ben-Or, Goldwasser, Kilian, and Wigderson [9]. However, the aforementioned works [22, 7, 20] refer to these proof systems by the generic term "oracle machine" (which refers to the syntax of the corresponding verifier). The term "probabilistically checkable proofs" was introduced in [4], and used ever since, although the term "locally verifiable (or testable) proofs" might have been much more appropriate (cf. [26, Sec. 13.2.2]).

## 6 Computationally Sound Proofs

The work of Micali [39] presented the notion of computationally-sound proof systems with relatively efficient proving procedures, termed *CS-proofs*. The notion of computationally-sound proofs (a.k.a arguments) was proposed before by Brassard, Chaum, and Crépeau [16], but in CS-proofs it is coupled with a relative-efficiency requirement (which refers to the completeness condition). Specifically, it is required that the complexity of proving valid statements be (polynomially) related to the complexity of determining the validity of the statement by one's own (i.e., without a proof). The abstract of the conference version of [39] reads as follows.

> This paper put forward a computationally-based notion of proof and explores its implications to computation at large.
>
> In particular, given a random oracle or a suitable cryptographic assumption, we show that every computation possesses a short certificate vouching its correctness, and that under a cryptographic assumption, any program for a NP-complete problem is checkable in polynomial time.
>
> In addition, our work provides the beginnings of a theory of computational complexity that is based on "individual inputs" rather than languages.

The construction presented by Micali [39] is similar to a previous construction of Kilian [36], but the fact that (unlike in [16, 36]) the notion of computational-soundness and the construction were de-coupled from zero-knowledge aspects helped focus attention on the notion and the construction.

Micali [39] also highlights the fact that CS-proof remain meaningful even if $\mathcal{P} = \mathcal{NP}$ and/or also when applied to decision problems in $\mathcal{P}$. Indeed, CS-proofs are related to doubly-efficient arguments, which are the computationally-sound variant of doubly-efficient interactive proof systems, which were introduced a decade and a half later by Goldwasser, Kalai, and Rothblum [31].

[6] The point is that the proof length is closely related to the randomness complexity: Specifically, a PCP of randomness complexity $r$ and query complexity $q$ uses proofs of ("effective") length at most $2^r \cdot q$.

# 7 Property Testing and its Connection to Learning and Approximation

The work of Goldreich, Goldwasser, and Ron [28] initiated a general study of Property Testing, while focusing on testing of graph properties (in the adjacency matrix representation). Property testing emerged, implicitly and before, in the work of Blum, Luby, and Rubinfeld [14]. The earlier line of work, focusing on algebraic properties, culminating in the work of Rubinfeld and Sudan [43], where the approach was abstracted and captured by the notion of a *robust characterization*, which corresponds to a special type of testers (i.e., non-adaptive testers of one-sided error probability). The work of Goldreich, Goldwasser, and Ron [28] advocated viewing property testing as a new type of computational problems, rather than as a tool towards program checking [13] (as viewed in [14]) or towards the construction of PCP systems (as in [6, 7, 20]). The abstract of the conference version of [28] reads as follows.

> We study the question of determining whether an unknown function has a particular property or is $\epsilon$-far from any function with that property. A property testing algorithm is given a sample of the value of the function on instances drawn according to some distribution, and possibly may query the function on instances of its choice.
>
> First, we establish some connections between property testing and problems in learning theory. Next, we focus on testing graph properties, and devise algorithms to test whether a graph has properties such as being $k$-colorable or having a $\rho$-clique (clique of density $\rho$ w.r.t the vertex set). Our graph property testing algorithms are probabilistic and make assertions which are correct with high probability, utilizing only poly$(1/\epsilon)$ edge-queries into the graph, where $\epsilon$ is the distance parameter. Moreover, the property testing algorithms can be used to efficiently (i.e., in time linear in the number of vertices) construct partitions of the graph which correspond to the property being tested, if it holds for the input graph.

As started in the original abstract, the main results of [28] are testers for a variety of graph partition problems all having query complexity that is independent of the size of the graph (but rather depending only on the proximity parameter).

In general, instances of the testing problems were viewed as descriptions of actual objects; that is, objects that arise from some application. Consequently, the representation of these objects as functions became a non-obvious step, which required justification. For example, in the case of testing graph properties, the starting point is the graph itself, and its representation as a function is an auxiliary conceptual step. In [28] graphs are represented by their adjacency relation (or matrix), which is not overly redundant when dense graphs are concerned, but in some subsequent works other alternatives were considered (see [26, Chap. 9-10]).

As hinted upfront, the notion of a tester presented in [28] allows for adaptive queries and two-sided error probability, while viewing non-adaptivity and one-sided error probability as special cases. While the bulk of their work [28, Sec. 5–10] focuses on testing graph properties, the paper also contains general results (see [28, Sec. 3-4]) and its definitional treatment (see [28, Sec. 2]) foresaw some directions that were pursued only in subsequent works. For more details on property testing see a recent textbook [26].

# 8 Pseudo-Deterministic Algorithms

The starting point of the work of Gat and Goldwasser [24] is the observation that probabilistic algorithms that solve search problem may output different solutions in different executions. That is, even if on input $x$ the algorithm outputs a correct solution with high probability (say with probability at least 2/3), it may be that no solution appears as output with significant probability (let alone with probability at least 2/3). Hence, their paper [24] initiates a study of search problems that may be solved in probabilistic polynomial-time by algorithms that, on each input $x$, output the same solution with probability at least 2/3. The abstract of their paper reads as follows.

> In this paper we introduce a new type of probabilistic search algorithm, which we call the *Bellagio* algorithm: a probabilistic algorithm which is guaranteed to run in expected polynomial time, and to produce a correct and *unique* solution with high probability. We argue the applicability of such algorithms for the problems of verifying delegated computation in a distributed setting, and for generating cryptographic public-parameters and keys in distributed settings. We exhibit several examples of Bellagio algorithms for problems for which no deterministic polynomial time algorithms are known. In particular, we show such algorithms for:
>
> - Finding a unique generator for $\mathsf{Z}_p$, when $p$ is a prime of the form $kq+1$ for $q$ is prime and $k = \text{polylog}(p)$. The algorithm runs in expected polynomial in $\log p$ time.
> - Finding a unique $q$'th non-residues of $\mathsf{Z}_p$ for any prime divisor $q$ of $p-1$, extending Lenstra's algorithm for finding unique quadratic non-residue of $\mathsf{Z}_p$. The algorithm runs in expected polynomial time in $\log p$ and $q$. The tool we use is a new variant of the Adleman-Manders-Miller probabilistic algorithm for taking $q$-th roots, which outputs a unique solution to the input equations and runs in expected polynomial time in $\log p$ and $q$.
> - Given a multi-variate polynomial $P \neq 0$, find a unique (with high probability) $x$ such that $P(x) \neq 0$. Alternatively you may think of this as producing a unique polynomial time verifiable certificate of inequality of polynomials.
>
> More generally, we show a necessary and sufficient condition for the existence of a Bellagio Algorithm for relation $R$: $R$ has a Bellagio algorithm if and only if it is deterministically reducible to some decision problem in BPP.

In later works (e.g., [30]) such algorithms were called *pseudodeterministic*, and the solution that they output, with high probability, was called *canonical*.

We stress that although most research in complexity theory refers to decision problems, search problems are at least as important. Recall that search problems are associated with binary relations, $R \subseteq \{0,1\}^* \times \{0,1\}^*$, and each element of $R(x) \stackrel{\text{def}}{=} \{y \in \{0,1\}^* : (x,y) \in R\}$ is called a solution to $x$ (and if $R(x) = \emptyset$ then $\perp$ is considered the only solution). Saying that $R$ can be solved by a randomized algorithm $A$ means that, for every $x$ that has a solution, it holds that $\Pr[A(x) \in R(x)] \geq 2/3$ (and $\Pr[A(x) = \perp] \geq 2/3$ if $R(x) = \emptyset$). Algorithm $A$ is called pseudodeterministic if for every $x$ there exists a (canonical) solution $s_x$ such that $\Pr[A(x) = s_x] \geq 2/3$.

The foregoing result of [24] asserts that $R$ can be solved by a pseudodeterministic polynomial-time algorithm if and only if solving $R$ is deterministically reducible in polynomial-time to some

*decision problem* in $\mathcal{BPP}$. In contrast, it was shown in [25] that for every $R$ that is recognizable in probabilistic polynomial-time, solving $R$ is deterministically reducible in polynomial-time to some *promise problem* in the promise class corresponding to $\mathcal{BPP}$. Hence, the difference between general randomized algorithms and pseudodeterministic algorithms is reflected in the difference between standard complexity classes (which refer to decision problems) and classes of promise problems.

We mention that the study of pseudodeterministic algorithms was recently extended to $\mathcal{RNC}$; in particular, finding perfect matchings in bipartite graphs (a problem known to be in $\mathcal{RNC}$ (but not in $\mathcal{NC}$)) was shown to have a pseudodeterministic NC algorithm [30].

## References


[1] L.M. Adleman and M. Huang. *Primality Testing and Abelian Varieties Over Finite Fields*. Springer-Verlag Lecture Notes in Computer Science (Vol. 1512), 1992. Preliminary version in *19th STOC*, 1987.

[2] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, Vol. 160 (2), pages 781–793, 2004.

[3] 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.

[4] 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.

[5] L. Babai. Trading Group Theory for Randomness. In *17th ACM Symposium on the Theory of Computing*, pages 421–429, 1985.

[6] L. Babai, L. Fortnow, and C. Lund. Non-Deterministic Exponential Time has Two-Prover Interactive Protocols. *Computational Complexity*, Vol. 1, No. 1, pages 3–40, 1991. Preliminary version in *31st FOCS*, 1990.

[7] 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.

[8] Z. Bar-Joseph and M. Ben-Or. A Tight Lower Bound for Randomized Synchronous Consensus. In *17th ACM Symposium on Principles of Distributed Computing*, pages 193–199, 1998.

[9] 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.

[10] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *20th ACM Symposium on the Theory of Computing*, pages 1–10, 1988.

[11] M. Ben-Or, E. Pavlov, and V. Vaikuntanathan. Byzantine agreement in the full-information model in $O(\log n)$ rounds. In *38th ACM Symposium on the Theory of Computing*, pages 179–186, 2006.

[12] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding. *SIAM Journal on Computing*, Vol. 36 (4), pages 889–974, 2006. Extended abstract in *36th STOC*, 2004.

[13] M. Blum and S. Kannan. Designing Programs that Check their Work. In *21st ACM Symposium on the Theory of Computing*, pages 86–97, 1989.

[14] M. Blum, M. Luby and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *Journal of Computer and System Science*, Vol. 47, No. 3, pages 549–595, 1993.

[15] G. Bracha. An O(log n) expected rounds randomized byzantine generals protocol. *Journal of the ACM*, Vol. 34 (4), pages 910–920, 1987. Preliminary version in *17th STOC*, 1985.

[16] 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.

[17] B. Chor and B.A. Coan. A Simple and Efficient Randomized Byzantine Agreement Algorithm. *IEEE Trans. Software Eng.*, Vol. 11 (6), pages 531–539, 1985. Preliminary version in *4th SRDS*, 1984.

[18] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, Vol. 17, pages 449–467, 1965.

[19] S. Even. *Graph Algorithms*. Computer Science Press, 1979. Second edition (edited by G. Even), Cambridge University Press, 2011.

[20] 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.

[21] P. Feldman and S. Micali. An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement. *SIAM Journal on Computing*, Vol. 26 (4), pages 873–933, 1997. Preliminary version in *16th ICALP*, 1989.

[22] L. Fortnow, J. Rompel and M. Sipser. On the power of multi-prover interactive protocols. In *3rd IEEE Symposium on Structure in Complexity*, pages 156–161, 1988. See errata in *5th IEEE Symposium on Structure in Complexity*, pages 318–319, 1990.

[23] H.N. Gabow. The Weighted Matching Approach to Maximum Cardinality Matching. *Fundamenta Informaticae*, Vol. 154 (1-4), pages 109–130, 2017.

[24] E. Gat and S. Goldwasser. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. In *ECCC*, TR11–136, 2011.

[25] O. Goldreich. In a World of P=BPP. In *Studies in Complexity and Cryptography*, Lecture Notes in Computer Science (Vol. 6650), Springer, pages 191–232, 2011.

[26] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[27] O. Goldreich and S. Goldwasser. On the Limits of Nonapproximability of Lattice Problems. *Journal of Computer and System Science*, Vol. 60 (3), pages 540–563, 2000. Preliminary version in *30th STOC*, 1998.

[28] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, pages 653–750, July 1998. Extended abstract in *37th FOCS*, 1996.

[29] 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.

[30] S. Goldwasser and O. Grossman. Bipartite Perfect Matching in Pseudo-Deterministic NC. In *44th ICALP*, pages 87:1–87:13, 2017.

[31] S. Goldwasser, Y. Kalai, and G.N. Rothblum. Delegating Computation: Interactive Proofs for Muggles. *Journal of the ACM*, Vol. 62(4), pages 27:1–27:64, 2015. Extended abstract in *40th STOC*, pages 113–122, 2008.

[32] S. Goldwasser and J. Kilian. Almost All Primes Can Be Quickly Certified. *Journal of the ACM*, Vol. 46 (4), pages 450–472, 1999. Preliminary version in *18th STOC*, 1986.

[33] 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. Earlier versions date to 1982.

[34] 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*, 1986.

[35] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, Vol. 2 (4), pages 225–231, 1973.

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

[37] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[38] 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.

[39] S. Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, Vol. 30 (4), pages 1253–1298, 2000. Preliminary version in *25th FOCS*, 1994.

[40] S. Micali and V.V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ Algorithm for Finding Maximum Matching in General Graphs. In *21st IEEE Symposium on Foundations of Computer Science*, pages 17–27, 1980.

[41] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, Vol. 27 (2), pages 228–234, 1980.

[42] V.R. Pratt. Every Prime Has a Succinct Certificate. *SIAM Journal on Computing*, Vol. 4 (3), pages 214–220, 1975.

[43] R. Rubinfeld and M. Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, Vol. 25(2), pages 252–271, 1996.

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

[45] V.V. Vazirani. A Proof of the MV Matching Algorithm. Unpublished manuscript, 2014. Available as `https://www.cc.gatech.edu/ vazirani/new-proof.pdf` (This is a revision of CoRR abs/1210.4594, 2012.)