# Derandomization in Cryptography

Boaz Barak*
Weizmann Institute of Science
Rehovot, Israel
boaz@wisdom.weizmann.ac.il

Shien Jin Ong†
Harvard University
Cambridge, MA, USA
shienjin@eecs.harvard.edu

Salil Vadhan‡
Harvard University
Cambridge, MA, USA
salil@eecs.harvard.edu

November 13, 2006

**Abstract**

We give two applications of Nisan–Wigderson-type ("non-cryptographic") pseudorandom generators in cryptography. Specifically, assuming the existence of an appropriate NW-type generator, we construct:

1. A one-message witness-indistinguishable proof system for every language in **NP**, based on any trapdoor permutation. This proof system does not assume a shared random string or any setup assumption, so it is actually an "**NP** proof system."
2. A noninteractive bit-commitment scheme based on any one-way function.

The specific NW-type generator we need is a hitting set generator fooling *nondeterministic circuits*. It is known how to construct such a generator if $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ has a function of nondeterministic circuit complexity $2^{\Omega(n)}$.

Our witness-indistinguishable proofs are obtained by using the NW-type generator to derandomize the ZAPs of Dwork and Naor (FOCS '00). To our knowledge, this is the first construction of an **NP** proof system achieving a secrecy property.

Our commitment scheme is obtained by derandomizing the interactive commitment scheme of Naor (J. Cryptology, 1991). Previous constructions of noninteractive commitment schemes were only known under incomparable assumptions.

**Keywords:** interactive proofs, witness-indistinguishable proofs, commitment schemes, complexity theory, pseudorandom generators.

*Supported by Clore Foundation Fellowship and Israeli Higher Education Committee Fellowship.

†Work done mainly while an undergraduate at MIT, supported by an MIT Eloranta Fellowship and the MIT Reed UROP Fund. Currently supported by ONR grant N00014-04-1-0478.

‡Supported by NSF grants CCR-0205423, CCR-0133096, and CNS-0430336, and a Sloan Research Fellowship.

# 1 Introduction

The computational theory of pseudorandomness has been one of the most fertile grounds for the interplay between cryptography and computational complexity. This interplay began when Blum, Micali, and Yao (BMY) [BM84, Yao82], motivated by applications in cryptography, placed the study of pseudorandom generators on firm complexity-theoretic foundations. They gave the first satisfactory definition of pseudorandom generators along with constructions meeting that definition. Their notion quickly acquired a central position in cryptography, but it turned out that the utility of pseudorandom generators was not limited to cryptographic applications. In particular, Yao [Yao82] showed that they could also be used for *derandomization* — efficiently converting randomized algorithms into deterministic algorithms. Pseudorandom generators and their generalization, pseudorandom functions [GGM86], also found a variety of other applications in complexity theory and the theory of computation (*e.g.,* [RR97, Val84]).

Focusing on derandomization, Nisan and Wigderson (NW) [NW94] proposed a weakening of the BMY definition of pseudorandom generators which still suffices for derandomization. The benefit was that such NW-type pseudorandom generators could be constructed under weaker assumptions than the BMY ones (circuit lower bounds for exponential time, rather than the existence of one-way functions).[1] Thus, a long body of work developed around the task of constructing increasingly efficient NW-type pseudorandom generators under progressively weaker assumptions. One of the highlights of this line of work is the construction of Impagliazzo and Wigderson [IW97] implying that $\mathbf{P} = \mathbf{BPP}$ under the plausible assumption that $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ has a problem of circuit complexity $2^{\Omega(n)}$. More recently, the work on NW-type pseudorandom generators has also been found to be intimately related to randomness extractors [Tre01], and has been used to prove complexity-theoretic results which appear unrelated to derandomization (*e.g.,* [IKW02]).

While allowing remarkable derandomization results such as the Impagliazzo–Wigderson result mentioned above, NW-type pseudorandom generators have not previously found applications in cryptography (for reasons mentioned below). In this work, we show that a stronger form of NW-type pseudorandom generators, namely ones fooling *nondeterministic circuits* [AK01, KvM02, MV05, SU05b], do have cryptographic applications. Using such pseudorandom generators (which can be constructed under plausible complexity assumptions), we:

1. Construct witness-indistinguishable "**NP** proofs" (i.e. one-message[2] proof systems, with a deterministic verifier strategy, and no shared random string or other setup assumptions) for every language in **NP**, assuming the existence of trapdoor permutations.

2. Construct *noninteractive* bit-commitment schemes from any one-way function.

Thus, each of these results requires two assumptions — the circuit complexity assumption for the NW-type pseudorandom generator (roughly, that **E** has a function of nondeterministic circuit complexity $2^{\Omega(n)}$) and a "cryptographic" assumption (one-way functions or trapdoor permutations).

Result 1 is the first construction of witness-indistinguishable **NP** proofs under any assumption whatsoever, and shows that interaction is not necessary to achieve secrecy in proof systems. It

[1] Here and throughout the paper, when we say that Assumption X is weaker than Assumption Y, we mean that Y is known to imply X but X is not known to imply Y. Strictly speaking, the assumptions for NW-type generators are only weaker in this sense when considering generators of the same stretch (and when fooling nonuniform circuits). In this paper, the NW-type generators we use have much greater stretch than the BMY-type generators we use, and hence the assumptions are incomparable.

[2] We use "messages" rather than "rounds," as the latter is sometimes used to refer to a pair of messages.

is obtained by derandomizing the ZAP construction of Dwork and Naor [DN00]. We note that Dwork and Naor [DN00] themselves also constructed one-message witness-indistinguishable proofs that are *nonuniform* in the sense that the prover and verifier require a polynomial-length string to be hardwired in advance as nonuniform advice. Those can be viewed as "**NP/poly** proofs".

Result 2 is not the first construction of noninteractive commitment schemes, but is based on assumptions that appear incomparable to previous ones (which were based on the existence of one-to-one one-way functions). We obtain this result by derandomizing the Naor's interactive bit-commitment scheme [Nao91].

These two examples suggest that NW-type pseudorandom generators (and possibly other "non-cryptographic" tools from the derandomization literature) are actually relevant to the foundations of cryptography, and it seems likely that other applications will be found in the future.

**NW-type Generators fooling Nondeterministic Circuits.** The most important difference between BMY-type and NW-type pseudorandom generators is that BMY-type pseudorandom generators are required to fool even circuits with greater running time than the generator, whereas NW-type pseudorandom generators are allowed greater running time than the adversarial circuit. Typically, a BMY-type pseudorandom generator must run in some fixed polynomial time (say $n^c$), and fool all polynomial-time circuits (even those running in time, say, $n^{2c}$). In contrast, an NW-type pseudorandom generator may run in time $n^{O(c)}$ (*e.g.,* $n^{3c}$) in order to fool circuits running in time $n^{2c}$. BMY-type pseudorandom generators are well-suited for cryptographic applications, where the generator is typically run by the legitimate parties and the circuit corresponds to the adversary (who is always allowed greater running time). In contrast, NW-type pseudorandom generators seem non-cryptographic in nature. Nevertheless we are able to use them in cryptographic applications. The key observation is that, in the protocols we consider, (some of) the randomness is used to obtain a string that satisfies some fixed property *which does not depend on the adversary (or its running time)*. Hence, if this property can be verified in some fixed polynomial time, we can obtain the string using an NW-type pseudorandom generator of a larger fixed polynomial running time. We then eliminate the randomness entirely by enumerating over all possible seeds. This is feasible because NW-type generators can have logarithmic seed length. Also, we show that in our specific applications, this enumeration does not compromise the protocol's security.

In the protocols we consider, the properties in question do not seem to be verifiable in polynomial time. However, they are verifiable in *nondeterministic* polynomial time. So we need to use a pseudorandom generator that fools nondeterministic circuits. Fortunately, it is possible for an *NW-type* pseudorandom generator to fool nondeterministic circuits, as realized by Arvind and Köbler [AK01] and Klivans and van Melkebeek [KvM02].[3] Indeed, a sequence of works have constructed such pseudorandom generators under progressively weaker complexity assumptions [AK01, KvM02, MV05, SU05b]. (Recently, these assumptions were all shown to be equivalent [SU05a]). Our results make use of the Miltersen–Vinodchandran construction [MV05] (which gives only a "hitting set generator" rather than a pseudorandom generator, but this suffices for our applications) and its "uniform analogue" from [GST03].

[3]It is impossible for a BMY-type pseudorandom generator to fool nondeterministic circuits, as such a circuit can recognize outputs of the pseudorandom generator by guessing the corresponding seed and evaluating the generator to check. Some attempts to bypass this difficulty can be found in [Rud97].

**Witness Indistinguishable NP Proofs.** In order to make zero-knowledge proofs possible, the seminal paper of Goldwasser, Micali, and Rackoff [GMR89] augmented the classical notion of an **NP** proof with two new ingredients — interaction and randomization. Both were viewed as necessary for the existence of zero-knowledge proofs, and indeed it was proven by Goldreich and Oren [GO94] that without either, zero-knowledge proofs exist only for trivial languages (those in **BPP**). The role of interaction was somewhat reduced by the introduction of "noninteractive" zero-knowledge proofs [BFM88, BDMP91], but those require a shared random string selected by a trusted third party, which can be viewed as providing a limited form of interaction. Given the aforementioned impossibility results [GO94], reducing the interaction further seems unlikely. Indeed, a truly noninteractive proof system, in which the prover sends a single proof string to the verifier, seems to be inherently incompatible with the intuitive notion of "zero knowledge": from such a proof, the verifier gains the ability to prove the same statement to others.

Despite this, we show that for a natural weakening of zero knowledge, namely *witness indistinguishability* [FS90], the interaction *can* be completely removed (under plausible complexity assumptions). Recall that a witness-indistinguishable proof system for a language $L \in \mathbf{NP}$ is an interactive proof system for $L$ that leaks no knowledge about which witness is being used by the prover (as opposed to leaking no knowledge at all, as in zero-knowledge proofs) [FS90]. Witness indistinguishability suffices for a number of the applications of zero knowledge [FS90], and also is a very useful intermediate step in the construction of zero-knowledge proofs [FLS99].

Several prior results show that witness-indistinguishable proofs do not require the same degree of interaction as zero-knowledge proofs. Feige and Shamir [FS90] constructed 3-message witness-indistinguishable proofs for **NP** (assuming the existence of one-way functions), whereas the existence of 3-message zero-knowledge proofs is a long-standing open problem. More recently, the ZAPs of Dwork and Naor [DN00] achieve witness indistinguishability with just 2 messages (assuming trapdoor permutations), whereas this is known to be impossible for zero knowledge [GO94]. As mentioned earlier, Dwork and Naor also showed that the interaction could be further reduced to one message at the price of *nonuniformity* (i.e. if the protocol can use some nonuniform advice of polynomial length); they interpret this as evidence that "*proving* a lower bound of two [messages] is unlikely."

We construct 1-message witness-indistinguishable proofs for **NP** in the "plain model", with no use of a shared random string or nonuniformity. Our proof system is obtained by derandomizing the verifier in the ZAPs of Dwork–Naor [DN00] via an NW-type generator against nondeterministic circuits. The prover, however, remains probabilistic polynomial-time given a witness for membership. Since our verifier is deterministic, we actually obtain a standard **NP** proof system with the witness indistinguishability property. More precisely, for any language $L \in \mathbf{NP}$ with associated **NP**-relation $R$, we construct a new **NP**-relation $R'$ for $L$. The relation $R'$ has the property that one can efficiently transform any witness with respect to $R$ into a distribution of new witnesses with respect to $R'$, such that the distributions originating from different witnesses of $R$ are computationally indistinguishable.

Converting **AM** proof systems to **NP** proof systems was actually one of the original applications of NW-type generators versus nondeterministic circuits [AK01, KvM02]. The novelty in our result comes from observing that this conversion preserves the witness indistinguishability property.

The randomness requirements of *zero-knowledge* proofs have been examined in previous works. Goldreich and Oren [GO94] showed that only languages in **BPP** have zero-knowledge proofs in which either the prover or verifier is deterministic. Thus De Santis, Di Crescenzo, and Persiano [DDP97, DDP99, DDP02] have focused on reducing the number of random bits. Specifically, under standard "cryptographic" assumptions, they constructed noninteractive zero-knowledge proofs with a shared random string of length $O(n^\varepsilon+\log(1/s))$ and 2-message witness-indistinguishable proofs (actually, ZAPs) in which the verifier uses only $O(n^\varepsilon + \log(1/s))$ random bits, where $\varepsilon > 0$ is any constant and $s$ is the soundness error. They posed the existence of 1-message witness-indistinguishable proofs for **NP** as an open problem. One of their main observations in [DDP02] is that combinatorial methods for randomness-efficient error reduction, such as pairwise independence and expander walks, preserve witness indistinguishability. As mentioned above, we make crucial use of an analogous observation about NW-type generators.

**Noninteractive Bit-Commitment Schemes.** Bit-commitment schemes are one of the most basic primitives in cryptography, used pervasively in the construction of zero-knowledge proofs [GMW91] and other cryptographic protocols. Informally, a bit-commitment scheme is a two-stage protocol between two interacting parties, the *sender* and the *receiver*. In the *commitment stage*, the sender commits to a secret bit $b$. In the *decommitment stage*, the sender decommits by revealing the bit $b$ together with an additional secret key that enables the receiver to verify that $b$ is consistent with the commitment stage. The commitment stage alone must not reveal any information about $b$. This is called the *hiding* property. In addition, we require that the commitment be *binding*, that is the sender should not be able to successfully decommit to more than one value.

Here we focus on perfectly (or statistically) binding and computationally hiding bit-commitment schemes; that is even computationally unbounded senders should not be able to decommit to different values, but the hiding property only holds against efficient (probabilistic polynomial time) receivers. *Noninteractive* bit-commitment schemes, in which the commitment phase consists of a single message from the sender to the receiver, are generally preferred over interactive schemes. There is a simple construction of noninteractive bit-commitment schemes from any *one-to-one* one-way function [Blu82, Yao82, GL89]. From general one-way functions, the only known construction of bit-commitment schemes, namely Naor's protocol [Nao91] with the pseudorandom generator construction of [HILL99], requires interaction.

We show how to use an NW-type pseudorandom generator against nondeterministic circuits to remove the interaction in Naor's protocol, yielding noninteractive bit-commitment schemes under assumptions that appear incomparable to the existence of one-to-one one-way functions. In particular, ours is a "raw hardness" assumption, not requiring hard functions with any structure such as being one-to-one.

From a different perspective, our result shows that "non-cryptographic" assumptions (nondeterministic circuit lower bounds for **E**) can reduce the gap between one-way functions and one-to-one one-way functions. In particular, a noninteractive bit-commitment scheme gives rise to a "partially one-to-one one-way function": a polynomial-time computable function $f(x, y)$ such that $x$ is uniquely determined by $f(x, y)$ and $x$ is hard to compute from $f(x, y)$ (for random $x, y$). It would be interesting to see if this can be pushed further to actually construct one-to-one one-way functions from general one-way functions under a non-cryptographic assumption.

**Perspective.** The assumption used in the construction of NW-type generators is a strong one, but it seems to be plausible (see Section 2.6). Perhaps its most significant feature is that it is very different than the assumptions typically used in cryptography (*e.g.,* it is a worst-case assumption); nevertheless, our results show it has implications in cryptography. In our first result, we use it to demonstrate the plausibility of 1-message witness-indistinguishable proofs for all of **NP**, which will hopefully lead to efficient constructions for specific problems based on specific assumptions. As for our second result, the plausibility of noninteractive commitment schemes was already established more convincingly based on one-to-one one-way functions [Blu82]. What we find interesting instead is that a "non-cryptographic" assumption can imply new relationships between basic cryptographic primitives, and in particular reduce the gap between one-way functions and one-to-one one-way functions.

# 2 Preliminaries

Let $X$ be a *random variable* taking values in a finite set $T$. We write $x \leftarrow X$ to indicate that $x$ is selected according to $X$. For a finite set $S$, we write $x \leftarrow S$ to indicate that $x$ is selected uniformly amongst all the elements of $S$.

We write $\mathrm{neg}(n)$ to denote a *negligible function*, namely one that vanishes more quickly than any inverse polynomial. That is, for all $c \in \mathbb{N}$, $\mathrm{neg}(n) < n^{-c}$ for all sufficiently large $n$. We write $\mathrm{poly}(n)$ to denote any polynomially bounded function; that is, $\mathrm{poly}(n) \leq n^c$ for some $c \in \mathbb{N}$, and for all sufficiently large $n$.

By *probabilistic polynomial time (PPT)*, we refer to probabilistic algorithms that run in *strict* polynomial time. A *nonuniform* PPT algorithm is a pair $(A, \bar{z})$, where $\bar{z} = z_1, z_2, \ldots$ is an infinite series of strings where $|z_n| = \mathrm{poly}(n)$, and $A$ is a PPT algorithm that receives pairs of input of the form $(x, z_{|x|})$. (The string $z_n$ is called the *advice string* for $A$ for inputs of length $n$.) Nonuniform PPT algorithms are equivalent to families of polynomial-sized Boolean circuits.

The *statistical difference* between two random variables $A$ and $B$ over $\{0,1\}^n$ is defined as

$$\Delta(A,B) \stackrel{\mathrm{def}}{=} \max_{T \subseteq \{0,1\}^n} |\Pr[A \in T] - \Pr[B \in T]| = \frac{1}{2} \sum_{x \in \{0,1\}^n} |\Pr[A \in T] - \Pr[B \in T]|.$$

We say that distributions $A$ and $B$ are *$\varepsilon$-close* if $\Delta(A,B) \leq \varepsilon$.

Let $I$ be a set of strings. A *probability ensemble* of a sequence of random variables indexed by $I$ is denoted as $\{A_x\}_{x \in I}$. Two probability ensembles $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are *computationally indistinguishable* on $I \subseteq \{0,1\}^*$ if for every PPT $D$, there exists a negligible function $\mu$ such that for all $x \in I$,

$$\Big|\Pr[D(x, A_x) = 1] - \Pr[D(x, B_x) = 1]\Big| \leq \mu(|x|).$$

We say that $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are *nonuniformly* computationally indistinguishable if the above holds for all *nonuniform* PPT $D$. Similarly, we say that $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are *statistically indistinguishable* if the above holds for all functions $D$ (instead of only PPT). Equivalently, $\{A_x\}_{x \in I}$ and $\{B_x\}_{x \in I}$ are statistically indistinguishable iff $A_x$ and $B_x$ are $\mu(|x|)$-close for some negligible function $\mu$ and all $x \in I$. Sometimes we will refer to ensembles indexed by natural numbers $n$, e.g. $\{A_n\}$ and $\{B_n\}$, in which case we apply the above definitions with $x = 1^n$. That is, we allow the distinguisher running time $\mathrm{poly}(n, |A_n|)$ and require the distinguishing gap to be negligible in $n$.

## 2.1 Nondeterministic Computations

A significant advantage of NW-type generators that we will use is that they can fool *nondeterministic* circuits, because even if such a circuit can guess the seed, it does not have enough time to evaluate the generator on it.

We define nondeterministic circuit to be a (nonuniform) Boolean circuit that has the additional power of nondeterminism.

**Definition 2.1.** A *nondeterministic* Boolean circuit $C(x, y)$ is a circuit that takes $x$ as its primary input and $y$ as a witness. For each $x \in \{0,1\}^*$, we define $C(x) = 1$ if there exist a witness $y$ such that $C(x, y) = 1$.

A *co-nondeterministic* Boolean circuit $C(x, y)$ is a circuit that takes $x$ as its primary input and $y$ as a witness. For each $x \in \{0,1\}^*$, we define $C(x) = 0$ if there exist a witness $y$ such that $C(x, y) = 0$.

Denote $S_{\mathrm{N}}(f)$ to be the minimal sized nondeterministic circuit computing $f$.

Nondeterministic and co-nondeterministic algorithms can be defined in a similar fashion, with the nonuniform circuit $C$ being replaced by a *uniform algorithm*. Naturally, we measure the running time of a nondeterministic algorithm $A(x, y)$ in terms of the first input $x$. Therefore **NP** and **coNP** are the classes of languages decidable by polynomial-time nondeterministic algorithms and co-nondeterministic algorithms, respectively.

**Definition 2.2.** A *nondeterministic* algorithm $A(x, y)$ is a uniform algorithm that takes $x$ as its primary input and $y$ as a witness. For each $x \in \{0,1\}^*$, we define $A(x) = 1$ if there exist a witness $y$ such that $A(x, y) = 1$.

Likewise, a *co-nondeterministic* algorithm $A(x, y)$ is a uniform algorithm that takes $x$ as its primary input and $y$ as a witness. For each $x \in \{0,1\}^*$, we define $A(x) = 0$ if there exist a witness $y$ such that $A(x, y) = 0$.

A nondeterministic (or co-nondeterministic) algorithm $A$ is said to run in time $t(n)$, if for every $x$ and $y$, the running time of $A(x, y)$ is at most $t(|x|)$.

## 2.2 Interactive Proofs

An interactive proof is an interactive protocol in which a prover (with unlimited computational powers) tries to convince a probabilistic polynomial-time verifier the validity of a certain statement. Since interactive protocols are probabilistic, the soundness and completeness criteria are also probabilistic. The formal definition of interactive proofs follows.

**Definition 2.3** (interactive proofs [BM88, GMR89])**.** An interactive protocol $(P, V)$ is called an *interactive proof system* for a language $L$ if the following conditions hold.

1. *(Efficiency)* On common input $x$, the number and total length of messages exchanged between $P$ and $V$ are bounded by a polynomial in $|x|$, and $V$ is a probabilistic polynomial-time machine.
2. *(Completeness)* If $x \in L$, then $\Pr[(P, V)(x) = 1] \geq \frac{2}{3}$.
3. *(Soundness)* If $x \notin L$, then for any $P^*$, $\Pr[(P^*, V)(x) = 1] \leq \frac{1}{3}$.

The class of languages possessing interactive proofs is denoted as **IP**.

We say that an interactive proof system has *perfect completeness* if the completeness condition holds with probability 1 instead of $\frac{2}{3}$. We say that a system has *perfect soundness* if the soundness condition holds with probability 0 instead of $\frac{1}{3}$.

An interactive proof system is called *public-coin* if the verifier's messages consist only of random strings and acceptance is computed as a deterministic polynomial-time function of the interaction's transcript. An interactive proof system that is not public-coin is called *private-coin*.

The number of *rounds* in an interactive proof is the total number of messages exchanged in the interaction (that is, both prover messages and verifier messages). A proof system with one round is called *noninteractive*.

### 2.3 The Class AM

The class **AM**, also known as Arthur-Merlin games, has two equivalent formulations. The first is as the class of languages with constant-message interactive proofs. The second is as the class of languages decidable by polynomial-time *probabilistic* nondeterministic algorithms. Formally, a probabilistic nondeterministic algorithm $A(x, r, y)$ takes a random input $r$ in addition to its regular input $x$ and nondeterministic input $y$. We say $A$ computes a function $f$ if the following two conditions hold.

1. If $f(x) = 1$, then $\Pr_r[\exists y A(x, r, y) = 1] = 1$.
2. If $f(x) = 0$, then $\Pr_r[\exists y A(x, r, y) = 1] \leq 1/2$.

Then **AM** is the class of languages decidable by such algorithms $A(x, r, y)$ running in time $\mathrm{poly}(|x|)$. The equivalence of the two definitions of **AM** is due to [BM88, GS89, FGM+89]. More generally, $\mathbf{AMTIME}(t(n))$ denotes the class of languages that are decided by probabilistic nondeterministic algorithms running in time $t(n)$, and $[\mathbf{i.o.}-\mathbf{AMTIME}](t(n))$ denotes the class of languages that are decided by probabilistic-time $t(n)$ nondeterministic algorithms for *infinitely many input lengths*. Formally, we say $L \in [\mathbf{i.o.}-\mathbf{AMTIME}](t(n))$ if there exists an algorithm $A$ running in time $t(n)$ such that for infinitely many $n \in \mathbb{N}$, the following two conditions hold for all $x$ of length $n$.

1. If $x \in L$, then $\Pr_r[\exists y A(x, r, y) = 1] = 1$.
2. If $x \notin L$, then $\Pr_r[\exists y A(x, r, y) = 1] \leq 1/2$.

Note that the above definition of $[\mathbf{i.o.}-\mathbf{AMTIME}](t(n))$ is slightly nonstandard in the sense that infinitely-often complexity classes are often defined in the following manner: If $\mathbf{C}$ is a complexity class, then $\mathbf{i.o.}\text{–}\mathbf{C}$ is the class of all languages $L$ such that there exists a language $L' \in \mathbf{C}$ such that $L \cap \{0,1\}^n = L' \cap \{0,1\}^n$ for infinitely many $n \in \mathbb{N}$. Observe that $\mathbf{i.o.}\text{–}\mathbf{AMTIME}(t(n)) \subseteq [\mathbf{i.o.}-\mathbf{AMTIME}](t(n))$. Discussions about the subtle difference between these two classes can be found in [GST03].

### 2.4 Pseudorandom Generators

A *pseudorandom generator* (PRG) is a deterministic algorithm $G\colon \{0,1\}^\ell \to \{0,1\}^m$, with $\ell < m$. Pseudorandom generators are used to convert a short random string into a longer string that looks random to any efficient observer.

**Definition 2.4** (Pseudorandom generator)**.** We say that $G\colon \{0,1\}^\ell \to \{0,1\}^m$ is a $(s, \varepsilon)$*-pseudorandom generator against circuits* if for all circuits $C\colon \{0,1\}^m \to \{0,1\}$ of size at most $s$, it holds that $|\Pr[C(G(U_\ell)) = 1] - \Pr[C(U_m) = 1]| < \varepsilon$, where $U_k$ denotes the uniform distribution over $\{0,1\}^k$.

**BMY-type vs. NW-type Generators.** As mentioned above, there are two main types of pseudorandom generators: Blum-Micali-Yao (BMY) [BM84, Yao82] type and Nisan-Wigderson (NW) [NW94] type generator. Both can be defined for a wide range of parameters, but here we focus on the "classic" settings that we need. A BMY-type generator is the standard kind of pseudorandom generator used in cryptography.

**Definition 2.5** (BMY-type generators)**.** A function $G = \bigcup_m G_m : \{0,1\}^\ell \to \{0,1\}^m$ is a *BMY-type pseudorandom generator* with seed length $\ell = \ell(m)$, if $G$ is computable in time $\mathrm{poly}(\ell)$, and for every constant $c$, $G_m$ is a $(m^c, 1/m^c)$-pseudorandom generator against circuits for all sufficiently large $m$.

Note that a BMY-type generator is required to have running time that is a fixed polynomial, but must fool circuits whose running time is an arbitrary polynomial. Håstad, Impagliazzo, Levin, and Luby [HILL99] proved that BMY-type pseudorandom generators with seed length $\ell(m) = m^\delta$ (for every $\delta > 0$) exist if and only if one-way functions exist.

NW-type generators differ from BMY-type generators most significantly in the fact that the generator has greater running time than the circuits it fools.

**Definition 2.6** (NW-type generators)**.** A function $G = \bigcup_m G_m : \{0,1\}^\ell \to \{0,1\}^m$ is an *NW-type pseudorandom generator* with seed length $\ell = \ell(m)$, if $G$ is computable in time $2^{O(\ell)}$ and $G_m$ is a $(m^2, 1/m^2)$-pseudorandom generator against circuits for all $m$.[4]

We will be interested in the "high end" NW-type generators, which have seed length $\ell(m) = O(\log m)$, and thus have running time which is a fixed polynomial in $m$. The running time of such a generator, though polynomial, is allowed to be greater than the size of the circuits it fools. Impagliazzo and Wigderson [IW97] proved that such a generator exists if $\mathbf{E} = \mathbf{DTIME}(2^{O(n)})$ has a function of circuit complexity $2^{\Omega(n)}$. Note that when the seed length is $\ell = O(\log m)$, all $2^\ell$ seeds can be enumerated in time $\mathrm{poly}(m)$, and hence the generator can be used for complete derandomization. In particular, the existence of such a generator implies that $\mathbf{BPP} = \mathbf{P}$.

## 2.5 Hitting Set Generators

A *hitting set generator* (HSG) is a deterministic algorithm $H(1^m, 1^s)$ that outputs a *set* of strings of length $m$. Hitting set generators are weaker notions of pseudorandom generators.

**Definition 2.7** (Hitting set generators)**.** We say that $H$ is an *$\varepsilon$-hitting set generator against circuits* if for every $m, s \in \mathbb{N}$, and circuit $C\colon \{0,1\}^m \to \{0,1\}$ of size at most $s$, the following holds.

$$\Pr[C(U_m) = 1] > \varepsilon \implies \exists y \in H(1^m, 1^s) \text{ such that } C(y) = 1.$$

Hitting set generators against *nondeterministic* and *co-nondeterministic* circuits are defined analogously, replacing circuits with nondeterministic and co-nondeterministic circuits respectively. In addition, we say that $H$ is an $\varepsilon$-hitting set generator against *co-nondeterministic uniform algorithms* if for every time-constructible function $s(\cdot)$ and every co-nondeterministic uniform algorithm $A\colon \{0,1\}^* \to \{0,1\}$ running in time at most $s(m)$ on inputs of length $m$, the following holds for all sufficiently large $m$.

$$\Pr[A(U_m) = 1] > \varepsilon \implies \exists y \in H(1^m, 1^{s(m)}) \text{ such that } A(y) = 1.$$

[4] One can replace $m^2$ in this definition with any *fixed* polynomial in $m$.

The construction of a one-message witness-indistinguishable proof system in Section 3 requires a hitting set generator against co-nondeterministic circuits. However, we will only need a (weaker) hitting set generator against co-nondeterministic uniform algorithms for the construction of a non-interactive commitment scheme in Section 4.

**Definition 2.8.** We say hitting set generator $H(1^m, 1^s)$ is *efficient* if its running time is polynomial in $m$ and $s$.

Note that the running time of an efficient hitting set generator, though a fixed polynomial, can be greater than the size of the circuits it fools. Hence, the hitting set generators defined correspond to NW-type generators. Furthermore, observe that a pseudorandom generator $G\colon \{0,1\}^\ell \to \{0,1\}^m$ fooling circuits of size $s$ gives rise to a hitting set generator, by taking the set of outputs of $G$ over all seeds. The hitting set generator will be efficient if $G$ is computable in time $\mathrm{poly}(s, m)$ and has logarithmic seed length $\ell = O(\log m + \log s)$. In this sense hitting set generators are weaker than pseudorandom generators. Indeed, hitting set generators can be directly used to derandomize algorithms with one-sided error (i.e. **RP** algorithms), whereas pseudorandom generators can be used to derandomize circuits with two-sided error (**BPP** algorithms). Since the error in **AM** proof systems can be made one-sided [FGM$^+$89], the existence of an efficient 1/2-HSG against co-nondeterministic circuits implies that $\mathbf{AM} = \mathbf{NP}$.

The first constructions of efficient HSG (in fact pseudorandom generators) against co-nondeterministic circuits was given by Arvind and Köbler [AK01]. Their construction was based on the assumption that there are languages in $\mathbf{E}$ that are hard on average for nondeterministic circuits of size $2^{\Omega(n)}$. Klivans and van Melkebeek [KvM02] gave a construction based on a *worst-case* hardness assumption; namely, the existence of languages in $\mathbf{E}$ with $2^{\Omega(n)}$ worst-case SAT-oracle circuit complexity, that is hardness for circuits with SAT-oracle gates. Miltersen and Vinodchandran [MV05] managed to relax the hardness condition to nondeterministic circuits (but only obtaining a hitting set generator rather than a pseudorandom generator). We state the main result of Miltersen and Vinodchandran [MV05].

**Theorem 2.9** ([MV05])**.** [5] *If there exist a function $f \in \mathbf{E}$ such that $S_{\mathrm{N}}(f) = 2^{\Omega(n)}$, then there exists an efficient 1/2-HSG against co-nondeterministic circuits. In particular, under this assumption* $\mathbf{AM} = \mathbf{NP}$.

Shaltiel and Umans [SU05b] subsequently extended Theorem 2.9 in two ways: First, they obtained a pseudorandom generator rather than a hitting set generator. Second, they obtained analogous results for quantitatively weaker assumption (e.g., when the $S_{\mathrm{N}}(f)$ is only superpolynomial rather than exponential) yielding correspondingly less efficient generators. However, we will not need these extensions in our paper.

**Uniform Hitting Set Generators.** Gutfreund, Shaltiel and Ta-Shma [GST03] extended Theorem 2.9 to give a hitting set generator against co-nondeterministic *uniform algorithms* from *uniform* hardness assumptions. They used the same hitting set generator as Miltersen and Vinodchandran, but augmented the analysis.

[5] [MV05] actually use a seemingly weaker assumption, only needing a function of exponential "single-valued" nondeterministic circuit complexity. But, it was recently shown in [SU05a] that all of the assumptions used in this line of work [AK01, KvM02, MV05] are in fact equivalent. In addition, [MV05] presented the HSG as a $(1-\delta)$-HSG for $\delta = 2^{m^\gamma}/2^m$, but it can be converted into a 1/2-HSG using dispersers as done implicitly in their paper.

**Theorem 2.10** ([GST03]). *If* $\mathbf{E} \not\subseteq [\mathbf{i.o.}-\mathbf{AMTIME}](2^{\delta n})$ *for some* $\delta > 0$*, then an efficient* $1/2$*-HSG against co-nondeterministic uniform algorithms exists.*

The assumption used in Theorem 2.10 is weaker than the assumption used in Theorem 2.9 since nonuniformity can be used to replace randomness.

### 2.6 Discussions

**Are the Assumptions Reasonable?** Our two results rely on the existence of hitting set generators as constructed in Theorems 2.9 and 2.10, which in turn make assumptions about $\mathbf{E}$ containing functions of high nondeterministic complexity. In our opinion, these assumptions are plausible. The two most common reasons to believe a hardness assumption are empirical evidence and philosophical (or structural) considerations. The widely held $\mathbf{P} \neq \mathbf{NP}$ assumption is supported by both. Empirically, much effort has been invested to finding efficient algorithms for $\mathbf{NP}$ problems. Philosophically, it seems unlikely that proofs should always be as easy to find as they are to verify. Other hardness assumptions, such as the hardness of factoring, are supported mainly by empirical evidence. Some, like $\mathbf{E} \not\subseteq \mathbf{NP}$ (equivalently, $\mathbf{EXP} \neq \mathbf{NP}$), are supported mainly by philosophical considerations: it seems unlikely that it should *always* be possible to prove the correctness of exponentially long computations with polynomial-sized proofs. The assumptions of Theorems 2.9 and 2.10 are natural strengthenings of this assumption, where we extend $\mathbf{NP}$ both by letting the running time grow from polynomial to subexponential and by allowing nonuniformity or randomization.

**How do we find the function $f$?** Once we accept the existence of *some* function $f \in \mathbf{E}$ such that $S_{\mathrm{N}}(f) = 2^{\Omega(n)}$, can we find a *specific* function $f$ satisfying that condition? The answer is yes. It is not hard to show that if there exists a function $f$ satisfying the condition of Theorem 2.9, then *every* function that is $\mathbf{E}$-complete via linear-time reductions also satisfies that condition. In particular, we can take the bounded halting function $\mathrm{BH}(\cdot)$ defined as follows: $\mathrm{BH}(M, x, t) = 1$ if the Turing machine $M$ outputs 1 on input $x$ after at most $t$ steps (where $t$ is given in binary), and $\mathrm{BH}(M, x, t) = 0$ otherwise.

## 3 Witness Indistinguishable NP Proofs

In this section we use efficient hitting set generators against co-nondeterministic circuits to derandomize the ZAP construction of Dwork and Naor [DN00] and obtain a *noninteractive witness indistinguishable* (WI) proof system for any language in $\mathbf{NP}$. We call this an "$\mathbf{NP}$ proof system" because it consists of a single message from the prover to the verifier, as is the case in the trivial $\mathbf{NP}$ proof of simply sending the witness to the verifier.

As in the trivial $\mathbf{NP}$ proof system, our verifier algorithm will be deterministic. Our prover algorithm, however, will be *probabilistic* to make our proof system witness indistinguishable. It remains open as to whether a probabilistic prover strategy is necessary to achieve the witness indistinguishability property. We stress that our proof system is in the *plain model*, without assumptions of a shared random string or nonuniformity. As far as we know, this is the first noninteractive proof system for $\mathbf{NP}$ in the plain model that satisfies a secrecy property.

## 3.1 Definitions

**Witness Relation.** Let $W \subseteq \{0,1\}^* \times \{0,1\}^*$ be a relation. Define $W(x) = \{w \mid (x,w) \in W\}$ and $L(W) = \{x \mid \exists w \text{ s.t. } (x,w) \in W\}$. If $w \in W(x)$ then we say that $w$ is a *witness* for $x$. Recall that the class **NP** is the class of languages $L$ such that $L = L(W)$ for a relation $W$ that is decidable in time polynomial in the first input. If $L = L(W)$ is an **NP** language then we say that $W$ is a *witness relation* corresponding to $L$.

**Efficient Provers.** Recall the notion of interactive proofs as defined in Section 2.2. Let $L$ be an **NP** language with witness relation $W$. We say that an interactive proof for $L$ has an *efficient prover* if the honest prover strategy can be implemented by a probabilistic polynomial-time algorithm given $w \in W(x)$ as auxiliary input. In this paper we will only be interested in interactive proofs for **NP** that have efficient provers.

**NP Proof Systems.** An **NP** *proof system* is an interactive proof system that is degenerate in that it (a) consists of only a single message from the prover to the verifier, (b) satisfies both perfect completeness and perfect soundness, and (c) has a deterministic verifier. The prover, however, is allowed to be probabilistic polynomial time given a witness of membership. Because the verifier is deterministic, an **NP** proof system for a language $L$ induces a witness relation $W$ corresponding to $L$ by setting $W(x)$ to contain all the prover messages accepted by the verifier.

**Witness Indistinguishability.** We recall the notion of witness indistinguishability (WI), as defined by Feige and Shamir [FS90].

**Definition 3.1** (witness indistinguishability, [FS90])**.** Let $L$ be an **NP** language with witness relation $W_L$. Let $(P,V)$ be a proof system for $L$ where $P$ is an efficient (probabilistic polynomial-time) prover that gets a witness as auxiliary input.

We say that $(P,V)$ is *witness indistinguishable (WI)* if for every nonuniform polynomial-time verifier $V^*$, every $x \in L$, and every $w, w' \in W_L(x)$, the interaction of $V^*$ with $P(w)$ is computationally indistinguishable from its interaction with $P(w')$. More precisely, the ensembles $\{\mathsf{output}_{V^*}(P(w),V^*)(x)\}_{x\in L,w,w'\in W_L(x)}$ and $\{\mathsf{output}_{V^*}(P(w'),V^*)(x)\}_{x\in L,w,w'\in W_L(x)}$ are (nonuniformly) computationally indistinguishable, where $\mathsf{output}_{V^*}(P(w),V^*)(x)$ is a random variable denoting the output of $V^*$ after interacting with $P$ when both receive common input $x$ and $P$ receives the witness $w$ as a private auxiliary input. Without loss of generality, $V^*$ outputs its *view* of the interaction, which consists of its coin tosses and the messages exchanged.

Feige and Shamir also proved that WI is closed under concurrent composition [FS90]. We will only need the special case of parallel composition, defined as follows. For an interactive proof system $(P,V)$ and a polynomial $m$, we define the *$m$-fold parallel execution* $(P_m(w),V_m)(x)$ to be the protocol whereby $P_m$ and $V_m$ execute $m(|x|)$ parallel copies of $(P(w),V)(x)$ with each party using independent coin tosses in each execution. Here 'parallel' means that the $i$'th message for each of the $m(|x|)$ executions are all sent simultaneously. Note that while the honest parties are defined to behave independently in each of the $m(|x|)$ executions, an adversary need not do so. Indeed, this is why zero knowledge is not preserved under parallel composition [GK96, FS90]. Nevertheless, witness indistinguishability is maintained:

**Theorem 3.2** ([FS90])**.** *If $(P,V)$ is witness indistinguishable, then $(P_m,V_m)$ is witness indistinguishable for every polynomial $m$.*

**ZAPs.** A *ZAP* [DN00] is a two-message public-coin interactive proof system that is witness indistinguishable. Dwork and Naor proved the following theorem.

**Theorem 3.3** ([DN00])**.** *If (nonuniformly secure) trapdoor permutations exist,*[6] *then every language in* **NP** *has a ZAP.*

We note that the construction of ZAPs by [DN00] is actually based on the weaker assumption that NIZK (noninteractive zero knowledge in the shared random string model) systems exist for every language in **NP**. Thus, our construction can also be based on this weaker assumption.

## 3.2 Our Result

The main theorem of this section follows.

**Theorem 3.4.** *Assume that there exists an efficient* $1/2$*-HSG against co-nondeterministic circuits and that (nonuniformly secure) trapdoor permutations exist. Then every language in* **NP** *has a witness-indistinguishable* **NP** *proof system.*

## 3.3 Proof of Theorem 3.4

We prove Theorem 3.4 by converting the ZAPs for languages in **NP** into witness indistinguishable **NP** proof systems. Let $L$ be an **NP** language with witness relation $W_L$, and let $(P, V)$ be the ZAP for $L$. We denote the first message in a ZAP (the verifier's random coins sent to the prover) by $r$ and denote the second message (sent by the prover to the verifier) by $\pi$. We let $\ell(n)$ denote the length of the verifier's first message in a proof for statements of length $n$. Let $x \in \{0,1\}^n \setminus L$. We say that $r \in \{0,1\}^{\ell(n)}$ is *sound* with respect to $x$ if there does not exist a prover message $\pi$ such that the transcript $(x, r, \pi)$ is accepting. The statistical soundness of the ZAP scheme implies that for every $x \in \{0,1\}^n \setminus L$, the probability that $r \leftarrow \{0,1\}^{\ell(n)}$ is sound with respect to $x$ is very high, and in particular it is larger than $\frac{1}{2}$.

Our construction is based on the following observation. Let $q(n)$ be a polynomial that bounds the running time of the honest ZAP verifier in a proof of statements of length $n$. For every $x \in \{0,1\}^n \setminus L$, there exists a *co-nondeterministic* circuit $C_x$ of size less than $p(n) < q(n)^2$ that outputs 1 if and only if a string $r$ is sound with respect to $x$. We stress that the time to verify the soundness of a string $r$ only depends on the running time of the honest verifier (in our case it is $p(n)$).

On input $r$, the circuit $C_x$ will output 1 if there does not exist a prover message $\pi$ such that the transcript $(x, r, \pi)$ is accepting, and 0 otherwise. Note that $\Pr[C_x(U_{\ell(n)}) = 1] > \frac{1}{2}$. Since $H$ is a $1/2$-HSG against co-nondeterministic circuits, we have that for every $x \in \{0,1\}^n \setminus L$, there exists $r \in H(1^{\ell(n)}, 1^{p(n)})$ such that $C_x(r) = 1$. In other words, for every $x \in \{0,1\}^n \setminus L$, there exists a string $r \in H(1^{\ell(n)}, 1^{p(n)})$ such that $r$ is sound with respect to $x$.

Our construction is as follows.

[6] We refer the reader to [Gol01, Sec. 2.4.4] for the definition of trapdoor permutations. Actually, the definition we use is what is called by Goldreich an *enhanced* trapdoor permutation collection. See discussion on [Gol04, Apdx. C.1]. Such a collection is known to exist based on either the RSA or factoring hardness assumptions [RSA78, Rab79].

**Protocol 3.5.** One-message WI **NP** proof for $L \in \mathbf{NP}$.

On common input $x \in \{0,1\}^n$ and auxiliary input $w$ for the prover, such that $(x, w) \in W_L$, do the following.

**Prover's message**

1. Compute $(r_1, \ldots, r_m) \stackrel{\text{def}}{=} H(1^{\ell(n)}, 1^{p(n)})$.
2. Using the auxiliary input (witness) $w$ and the ZAP prover algorithm, compute for every $i \in [1, m]$, a string $\pi_i$ that is the prover's response to the verifier's message $r_i$ in a ZAP proof for $x$.
3. Send to verifier $(\pi_1, \ldots, \pi_m)$.

**Verifier's Test**

1. Compute $(r_1, \ldots, r_m) \stackrel{\text{def}}{=} H(1^{\ell(n)}, 1^{p(n)})$.
2. Given prover's message $(\pi_1, \ldots, \pi_m)$, run the ZAP verifier on the transcript $(x, r_i, \pi_i)$, for every $i \in [1, m]$.
3. Accept if the ZAP verifier accepts *all* these transcripts.

Note that Protocol 3.5 is indeed a one-message system with a deterministic verifier, and it satisfies the perfect completeness property. Thus, to prove Theorem 3.4, we need to prove that it has perfect soundness and is witness indistinguishable.

**Lemma 3.6.** *Protocol 3.5 is a perfectly sound proof system for L.*

*Proof.* Let $x \notin L$, with $|x| = n$. Since $H$ is a HSG, there exists an $r_i \in H(1^{\ell(n)}, 1^{p(n)})$ that is sound with respect to $x$. This means that no prover's message $\pi_i$ will make the ZAP verifier accept the transcript $(x, r_i, \pi_i)$. Therefore, no string $\pi = (\pi_1, \ldots, \pi_m)$ will make the verifier of Protocol 3.5 accept. $\square$

**Lemma 3.7.** *Protocol 3.5 is a witness indistinguishable (WI) proof system for L.*

*Proof.* This follows from the fact that the prover algorithm of Protocol 3.5 simply invokes $m$ times the prover algorithm for the $ZAP$ on $m$ different verifier messages. Since the $m$-fold parallel composition of the ZAP is also witness indistinguishable (by Theorem 3.2), it follows that Protocol 3.5 is witness indistinguishable. More precisely, the output (or view) of the verifier after an execution of Protocol 3.5 where the prover uses witness $w$ can be simulated perfectly by a cheating verifier $V_m^*$ that sends the $m$-tuple of messages $(r_1, \ldots, r_m) = H(1^{\ell(n)}, 1^{p(n)})$ in the $m$-fold parallel composition $(P_m(w), V_m^*)$. Since the output of $V_m^*$ is computationally indistinguishable for any two witnesses used by the prover $P_m$, so is the output of the verifier in Protocol 3.5. $\square$

## 3.4 Applications of Noninteractive WI Proofs

**1-out-of-2 Oblivious Transfer.** As an application of ZAPs, Dwork and Naor [DN00] constructed a *3-message* 1-out-of-2 *oblivious transfer* (OT) protocol based on the Quadratic Residuosity Assumption.[7] Informally, an OT protocol consists of two parties, a sender and a receiver. The sender has two secret input bits $b_0$ and $b_1$. The goal of the receiver is to select an input bit of the sender without letting the sender know which bit it had selected. The goal of the sender is to allow the chooser to learn only its selected input bit.

The first two rounds of the Dwork-Naor OT protocol consist of a ZAP (2-message WI proof) of a certain **NP** statement. Replacing the ZAP with our WI **NP** proofs, we prove that same **NP** statement in only one message, thus allowing for a *2-message* OT with the same security properties.

We begin with the formal definition of OT that we use. Let $\mathsf{output}_S(S(b_0, b_1; r_S), R(c, r_R))$ denote the output of sender $S$ (on inputs $b_0$ and $b_1$, and private randomness $r_S$) after interacting with receiver $R$ (on inputs the choice bit $c$ and private randomness $r_R$). Both parties are also given the security parameter $k$, but we omit it from the notation. We define $\mathsf{output}_R(S(b_0, b_1; r_S), R(c, r_R))$ in an analogous manner.

**Definition 3.8.** An *1-out-of-2 oblivious transfer (OT) protocol* (with security parameter $k$) consists of a polynomial-time sender $S$ and a polynomial-time receiver $R$, satisfying the following conditions.

1. *(Completeness)* For all $b_0, b_1, c \in \{0, 1\}$, we have that $\Pr_{r_S, r_R}[\mathsf{output}_R(S(b_0, b_1; r_S), R(c, r_R)) = b_c] > 1 - \mathrm{neg}(k)$.
2. *(Computational privacy of receiver)* For all probabilistic polynomial-time cheating $S^*$, we have that $\mathsf{output}_{S^*}(S^*, R(0; r_R))$ is computationally indistinguishable from $\mathsf{output}_{S^*}(S^*, R(1; r_R))$.
3. *(Statistical privacy of sender)* For every deterministic receiver strategy $R^*$, one of the two following conditions holds:
   (a) $\mathsf{output}_{R^*}(S(0, b; r_S), R^*)$ is statistically indistinguishable from $\mathsf{output}_{R^*}(S(1, b; r_S), R^*)$ for every $b \in \{0, 1\}$, or
   (b) $\mathsf{output}_{R^*}(S(b, 0; r_S), R^*)$ is statistically indistinguishable from $\mathsf{output}_{R^*}(S(b, 1; r_S), R^*)$ for every $b \in \{0, 1\}$.

Condition 3 intuitively says that the receiver obtains no information about at least one of the sender's inputs. Unlike simulation-based definitions, however, it does not guarantee that a cheating receiver "knows" which of the two inputs it is learning. Similar definitions have been used in previous works on OT with few rounds.

As mentioned above, we obtain a 2-message OT protocol by using noninteractive WI proofs in the Dwork–Naor [DN00] protocol. The computational assumptions we make are the existence of HSG against co-nondeterministic circuits and the Quadratic Residuosity Assumption, the latter being inherited from [DN00]. (We can drop the assumption of trapdoor permutations in Theorem 3.4, because it is implied by the Quadratic Residuosity Assumption.) The formal theorem is stated below.

**Theorem 3.9.** *Suppose that there exists an efficient 1/2-HSG against co-nondeterministic circuits and that the Quadratic Residuosity Assumption holds. Then there exists a* 2-message *1-out-of-2 OT protocol.*

There are two points that we would like to note. First, our protocol does *not* use any *public key.* If we allow the sender to publish a public key, the Dwork-Naor OT protocol can be reduced

[7] For further information on the Quadratic Residuosity Assumption, we refer the reader to [GB01, Section 2.5.1].

to two messages by having the sender $S$ publish the random string of the ZAP in the public key (this random string corresponds to the first message of the ZAP).

Second, there were several previous constructions of 2-message OT protocols satisfying similar security properties as Definition 3.8. Naor and Pinkas [NP01] and Aiello, Ishai, and Reingold [AIR01] independently constructed 2-message OT protocols based on the Decisional Diffie–Hellman (DDH) Assumption. Recently and independently of our work, Kalai [Kal05] constructed 2-message OT protocols based on a variant of "smooth projective hash families" [CS02].

**Weak Zero Knowledge.** The standard notions of zero knowledge require at least three rounds of interaction for languages outside **BPP** [GO94, BLV06]. Subsequent to this work, Barak and Pass [BP04] proposed a weak form of zero-knowledge protocols for all languages in **NP** that consist only of a *single* round, *i.e.,* a single message from the prover to the verifier. Like our 1-message WI, their prover is randomized and their verifier is deterministic. Their construction of such protocols utilizes noninteractive WI proofs, as constructed in this paper. The properties achieved are weaker in the following sense: The weak zero-knowledge condition allows the simulator to run in *quasi-polynomial* time instead of polynomial time, and the computational soundness is only guaranteed against *uniform* probabilistic polynomial-time cheating provers.

# 4 Noninteractive Bit Commitment

*Bit-commitment schemes* are basic primitives in cryptography. Informally, a bit-commitment scheme is a protocol that consists of two interacting parties, the sender and the receiver. The first step of the protocol involves the sender giving the receiver a commitment to a secret bit $b$. In the next step, the sender decommits the bit $b$ by revealing a secret key. The commitment alone (without the secret key) must not reveal any information about $b$. This is called the *hiding* property. In addition, we require that the commitment to $b$ be *binding*, that is the sender should not be able to decommit to a different bit $\bar{b}$. Note that given a bit-commitment scheme, a string-commitment scheme can be obtained by independently committing to the individual bits of the string (cf., [Gol01]).

In an *interactive bit-commitment scheme*, the sender and the receiver are allowed to interact during the commitment and decommitment steps. The formal definition of an interactive bit-commitment scheme can be found in [Gol01]. Often, however, *noninteractive* bit-commitment schemes are preferred or even crucial. For these, a simpler definition can be given.

**Definition 4.1** (noninteractive bit commitment)**.** A *noninteractive bit-commitment scheme* is a polynomial-time algorithm $S$ which takes a bit $b \in \{0,1\}$ and a random key $K \leftarrow \{0,1\}^{\mathrm{poly}(k)}$, where $k$ is the security parameter, and outputs a commitment $C = S(b;K)$. The algorithm $S$ must satisfy the following two conditions:

1. (Binding) There do not exist keys $K, K'$ such that $S(0;K) = S(1;K')$.

2. (Hiding) The commitments to 0 and 1 are computationally indistinguishable. This means that the probability distributions $\{S(0;K)\}_{K \leftarrow \{0,1\}^{\mathrm{poly}(k)}}$ and $\{S(1;K)\}_{K \leftarrow \{0,1\}^{\mathrm{poly}(k)}}$ are computationally indistinguishable by probabilistic polynomial-time algorithms.

We say that a bit-commitment scheme is *nonuniformly* secure if the probability distributions $\{S(0;K)\}_{K \leftarrow \{0,1\}^{\mathrm{poly}(k)}}$ and $\{S(1;K)\}_{K \leftarrow \{0,1\}^{\mathrm{poly}(k)}}$ are nonuniformly computationally indistinguishable. This means that even nonuniform polynomial-sized circuits cannot distinguish between a commitment to 0 and a commitment to 1.

There is a well-known construction by Blum [Blu82] of a noninteractive bit-commitment scheme based on any *one-to-one* (1-1) one-way function (using the function's hard-core predicate [Yao82, GL89]). Naor [Nao91] gave a construction of an *interactive* bit-commitment scheme based on any one-way function (using pseudorandom generators [HILL99]).

For completeness, we briefly describe a noninteractive bit commitment protocol based on any 1-1 one-way function. First, note that a 1-1 one-way function can be transformed into another 1-1 one-way function with an associated hard-core predicate [Yao82, GL89]. Then, let $f$ be a 1-1 one-way function and let $h$ be the hard-core predicate for $f$. A commitment to a bit $b \in \{0,1\}$ is just $\langle f(K), h(K) \oplus b \rangle$, where $K$ is a randomly chosen key. The injectivity property of $f$ seems crucial to guarantee the binding property of the commitment scheme.

## 4.1 Our Result

The main result of this section is the following theorem.

**Theorem 4.2.** *Assume that there exists an efficient* $1/2$*-HSG against co-nondeterministic* uniform *algorithms. Then, noninteractive bit-commitment schemes exist if and only if one-way functions exist.*

The first condition is true if $\mathbf{E} \not\subseteq [\mathbf{i.o.}{-}\mathbf{AMTIME}](2^{\Omega(n)})$, by Theorem 2.10. We stress that the assumption of an efficient 1/2-HSG against co-nondeterministic *uniform* algorithms is sufficient, even if one wants to obtain a commitment scheme that is nonuniformly secure (*i.e.,* commitments that are indistinguishable by polynomial-sized circuits). However, to get such schemes it will be necessary to assume that the one-way function is secure against *nonuniform* polynomial-sized circuits.

If we assume that the one-way function is only secure against uniform probabilistic polynomial-time adversaries, then we obtain commitment schemes secure against (uniform) probabilistic polynomial-time algorithms.

Our result is incomparable to the previous results on bit-commitment schemes. This is because the assumptions used in constructing our noninteractive commitments are on one hand stronger than Naor's [Nao91], which only requires one-way functions, but Naor's scheme is interactive. On the other hand, our assumptions seems incomparable to assuming the existence of 1-1 one-way functions.

**"Raw" Hardness vs. Hardness with Structure.** Note that unlike assuming the existence of 1-1 one-way functions, we do not assume in Theorem 4.2 that there exists a hard function with a particular structure. Rather, we only assume that there exists functions with "raw hardness" (i.e., a one-way function and a function in $\mathbf{E}$ with high $\mathbf{AM}$-complexity).

Even if one is told that one-to-one one-way functions exist, it is necessary to know a *particular* one-to-one one-way function to instantiate Blum's noninteractive commitment scheme. In contrast, we can construct a single noninteractive commitment scheme that is secure as long as there exists a one-way-function and a function $f \in \mathbf{E} \setminus [\mathbf{i.o.}{-}\mathbf{AMTIME}](2^{\Omega(n)})$. This is because we can instantiate our scheme with a universal one-way function[8] [Lev87] and a function that is **E**-complete via linear-time reductions such as the function $\mathrm{BH}(\cdot)$ (see discussion in Section 2.6).

## 4.2 Proof of Theorem 4.2

Our construction is based on derandomizing Naor's [Nao91] *interactive* bit-commitment scheme using a hitting set generator.

Let $G\colon \{0,1\}^k \to \{0,1\}^{3k}$ be BMY-type pseudorandom generator computable in time $k^d$ for some constant $d$. Such a generator can be constructed based on any one-way function [HILL99]. Naor [Nao91] gave the following protocol for an interactive bit-commitment scheme, based on the existence of such a generator.

**Protocol 4.3.** Interactive bit-commitment scheme [Nao91].

Input to receiver $R$: $1^k$, where $k$ is the security parameter.
Input to sender $S$: $1^k$ and a bit $b \in \{0,1\}$.

**Commitment stage:**

**R:** Select a random $r \leftarrow \{0,1\}^{3k}$ and send $r$ to $S$.

**S:** Select a random $s \leftarrow \{0,1\}^k$. If $b = 0$, send $\alpha = G(s)$ to $R$. Else, if $b = 1$, send $\alpha = G(s) \oplus r$ to $R$.

**Decommitment stage:**

**S:** Reveal $s$ and $b$.

**R:** Accept if $b = 0$ and $\alpha = G(s)$, or $b = 1$ and $\alpha = G(s) \oplus r$.

Observe that when the sender commits to 0, the sender's message $\alpha$ is distributed according to $G(U_k)$. When the sender commits to 1, $\alpha$ is distributed according to $G(U_k) \oplus r$. The following lemma, shows that Protocol 4.3 has the hiding property.

**Lemma 4.4** (hiding property)**.** *For every $r \in \{0,1\}^{3k}$, the distributions $G(U_k)$ and $G(U_k) \oplus r$ are computationally indistinguishable.*[9]

*Proof.* For any efficient adversary $A$, the pseudorandomness of $G$ guarantees that

$$|\Pr[A(G(U_k)) = 1] - \Pr[A(U_{3k}) = 1]| < \varepsilon,$$

and for any given $r \in \{0,1\}^{3k}$,

$$|\Pr[A(G(U_k) \oplus r) = 1] - \Pr[A(U_{3k}) = 1]| < \varepsilon',$$

[8] The construction of such a universal one-way function can also be found in [Gol01][Sec. 2.4.1]. It uses the observation that if there exists a one-way-function, then there exists a one-way function that is computable in time $n^2$.

[9] To be exact, the condition of the lemma ("for every $r$") holds for nonuniform computational indistinguishability (assuming that $G$ is a pseudorandom generator against nonuniform circuits). For the uniform setting, the lemma still holds if the string $r$ comes from any polynomial-time samplable distribution. That is, $r \leftarrow R(1^k)$, where $R$ is a probabilistic polynomial-time algorithm.

where $\varepsilon$ and $\varepsilon'$ are negligible. Hence by the triangle inequality,

$$|\Pr[A(G(U_k) \oplus r) = 1] - \Pr[A(G(U_k)) = 1]| < \varepsilon + \varepsilon' = \text{neg}(k).$$

This shows that no efficient adversary $A$ can distinguish between $G(U_k)$ and $G(U_k) \oplus r$. $\square$

Define a string $r \in \{0,1\}^{3k}$ to be *good* for $G$ if for all $s, s' \in \{0,1\}^k$, we have $G(s) \neq G(s') \oplus r$. We have the following lemma.

**Lemma 4.5** (binding property)**.** *The probability* $\Pr_{r \leftarrow \{0,1\}^{3k}}[r \textit{ is } \text{good}] \geq 1 - 2^{-k}$.

*Proof.* Note that $G(s) \neq G(s') \oplus r$ iff $G(s) \oplus G(s') \neq r$. The total number of pairs $(s, s')$, with $s, s' \in \{0,1\}^k$, is $2^{2k}$. For each pair, only one $r$ is not good, namely $r = G(s) \oplus G(s')$. Hence, the number of $r \in \{0,1\}^{3k}$ which are not good is at most $2^{2k}$. This implies that the fraction of good $r \in \{0,1\}^{3k}$ is at least $1 - 2^{2k}/2^{3k} = 1 - 2^{-k}$. $\square$

If the receiver selected a good $r$ in the first step of the commitment stage of Protocol 4.3, then there do not exist $s, s' \in \{0,1\}^k$ such that $G(s) = G(s') \oplus r$, so no commitment $\alpha$ can be opened as both a 0 and 1. The probability of selecting a good $r$ is high, hence Protocol 4.3 is binding.

### 4.2.1 Our Noninteractive Bit-Commitment Scheme.

Observe that the only interaction involved in Protocol 4.3 is in the receiver sending a random $r \in \{0,1\}^{3k}$ to the sender. However, one can see that the receiver does not have to send a random string, and it is enough to send a *good* string. This is because a good string $r$ will make the distributions $G(U_k)$ and $G(U_k) \oplus r$ disjoint. As we show in the proof of Lemma 4.8, testing whether $r$ is good can be done by a (uniform) *co-nondeterministic* algorithm running in time at most $3k^d$. Since the fraction of good $r$'s is large, an efficient HSG against co-nondeterministic algorithms $H$ can be used to select a candidate list of $r$'s such that at least one element $r \in H$ is good. Thus, our protocol will be obtained by running the sender of Naor's protocol on each $r$ in the hitting set. The resulting protocol follows.

**Protocol 4.6.** Noninteractive bit-commitment scheme.

Input to receiver $R$: $1^k$, where $k$ is the security parameter.
Input to sender $S$: $1^k$ and a bit $b \in \{0,1\}$.

**Commitment stage:**

1. Compute $\big(r_1, \ldots, r_{p(k)}\big) \stackrel{\text{def}}{=} H(1^{3k}, 1^{3k^d})$.
2. Choose $s_1, \ldots, s_{p(k)}$ at random from $\{0,1\}^k$.
3. If $b = 0$, send $\alpha = \langle G(s_1), \ldots, G(s_{p(k)})\rangle$.
   If $b = 1$, send $\alpha = \langle G(s_1) \oplus r_1, \ldots, G(s_{p(k)}) \oplus r_{p(k)}\rangle$.

**Decommitment stage:**
$S$ reveals $b$ and $\langle s_1, \ldots, s_{p(k)}\rangle$. $R$ accepts if either of the following holds:

1. The bit $b = 0$ and $\alpha = \langle G(s_1), \ldots, G(s_{p(k)})\rangle$.
   or
2. The bit $b = 1$ and $\alpha = \langle G(s_1) \oplus r_1, \ldots, G(s_{p(k)}) \oplus r_{p(k)}\rangle$.

To show that Protocol 4.6 constitutes a bit-commitment scheme (and hence to prove Theorem 4.2), we prove the following two lemmas.

**Lemma 4.7.** *Protocol 4.6 has the hiding property of Definition 4.1.*

*Proof.* By Lemma 4.4, we know that for an $r \in \{0,1\}^{3k}$ generated by a polynomial-time algorithm, the distributions $G(U_k)$ and $G(U_k) \oplus r$ are computationally indistinguishable. Furthermore given $r$, the distributions $G(U_k)$ and $G(U_k) \oplus r$ are polynomial-time samplable. Hence, by a hybrid/statistical-walk argument, the distributions $\langle G(U_k^1), G(U_k^2), \ldots, G(U_k^{p(k)})\rangle$ and $\langle G(U_k^1) \oplus r_1, G(U_k^2) \oplus r_2, \ldots, G(U_k^{p(k)}) \oplus r_{p(k)}\rangle$ are computationally indistinguishable, for $\big(r_1, r_2, \ldots, r_{p(k)}\big) = H(1^{3k}, 1^{3k^d})$. □

**Lemma 4.8.** *Protocol 4.6 has the binding property of Definition 4.1.*

*Proof.* Define the co-nondeterministic (uniform) algorithm $A$ such that $A(r) = 1$ if $\forall s, s'\ G(s) \oplus G(s') \neq r$. Note that $A(r) = 1$ if and only if $r$ is good. Therefore $\Pr[A(U_{3k}) = 1] \geq 1 - 2^{-k} > 1/2$. In addition, the running time of $A$ (on inputs of length $k$) is bounded by $3k^d$. Hence, there exists an $r_i \in H(1^{3k}, 1^{3k^d})$ such that $\forall s, s'\ G(s) \oplus G(s') \neq r_i$. Therefore, there do not exist $s_1, \ldots, s_{p(k)}$ and $s'_1, \ldots, s'_{p(k)}$ such that

$$\langle G(s_1), \ldots, G(s_{p(k)})\rangle = \langle G(s'_1) \oplus r_1, \ldots, G(s'_{p(k)}) \oplus r_{p(k)}\rangle.$$

In other words, no commitment $\alpha$ can be opened as both a 0 and 1. Thus, Protocol 4.6 is perfectly binding. □

If one-way functions exists, then pseudorandom generators exist [HILL99] and hence noninteractive bit-commitment schemes exists. Conversely, noninteractive bit-commitment schemes (as in Definition 4.1) imply the existence of one-way functions [IL89]. These facts, together with Lemmas 4.7 and 4.8 establish Theorem 4.2.

### 4.3 Partially One-to-one One-way Functions

Another interpretation of our result is as closing the gap between one-to-one and general one-way functions under a "non-cryptographic" assumption.

**Definition 4.9.** A function $f = \cup_k f_k \colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}^*$ is a *partially one-to-one one-way function* if

1. (easy to evaluate) $f$ can be evaluated in polynomial time.
2. (partially one-to-one) If $f(x,y) = f(x',y')$, then $x = x'$.
3. (hard to invert) For every probabilistic polynomial-time algorithm $A$, $\Pr\left[A(1^k, f(X,Y)) = X\right]$ is negligible in $k$, where the probability is taken over $X$ and $Y$ chosen independent and uniformly from $\{0,1\}^k$, and the coin tosses of $A$.

**Lemma 4.10.** *Partially one-to-one one-way functions exist iff noninteractive bit-commitment schemes exist.*

*Proof.* If $f$ is a partially one-to-one one-way function, we can obtain a noninteractive bit-commitment scheme using the Goldreich–Levin hardcore bit [GL89]. Specifically, define $\texttt{Commit}(b; x, y, r) = (f(x,y), r, \langle x, r\rangle \oplus b)$, where $\langle \cdot, \cdot \rangle$ denotes inner product mod 2.

If a noninteractive bit-commitment scheme exists, we can obtain a partially one-to-one one-way function by first converting the bit-commitment scheme to a string-commitment scheme (by committing independently to each bit) and then defining $f(x,y) = \texttt{Commit}(x; y)$. (The fact that $x$ and $y$ may be of different lengths is inconsequential, and can be fixed by padding.) □

Thus, a restatement of Theorem 4.2 is the following.

**Corollary 4.11.** *Assume that there exists an efficient* $1/2$*-hitting set generator against co-nondeterministic uniform algorithms. Then one-way functions imply partially one-to-one one-way functions.*

## 5 Future Work

Given the two examples we have presented here, it is natural to look for more applications of NW-type generators (and related notions in complexity theory) to cryptography. In parallel to this work, Barak, Lindell, and Vadhan [BLV06] have used NW-type generators to obtain *negative* results about zero-knowledge proofs.

To facilitate the search for additional applications, we summarize the properties of the protocols (ZAPs and Naor's bit commitment) that enabled our derandomizations to work.

1. In order for the protocol to be secure, the random string $r$ need only satisfy some fixed property that depends on only the algorithms of the "honest parties". In particular, it should be possible to verify this property by a nondeterministic algorithm that runs in a fixed polynomial time. (Algorithms even higher in the polynomial hierarchy can also be derandomized under stronger complexity assumptions [KvM02].)
2. The protocol must remain secure under parallel repetition (with multiple choices of $r$, at least one of which satisfies the property above).

Another intriguing question is whether it can be shown that under a "non-cryptographic" assumption, one-way functions imply truly one-to-one one-way functions (rather than just partially one-to-one ones).

Finally, given our plausibility result, it is natural to look for additional constructions of nontrivial one-message witness-indistinguishable proofs. Either constructions for specific problems based on specific assumptions or general constructions for all of **NP** based on alternative assumptions would be interesting. In addition to complexity-theoretic assumptions, it may also be useful to use assumptions from number theory, such as the Extended Riemann Hypothesis, which has been used for derandomization in the past (e.g. [Mil76]).

**Acknowledgments.**

We thank the anonymous We thank Oded Goldreich and the anonymous SICOMP and CRYPTO 2003 reviewers for helpful comments.

## References

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, pages 119–135, 2001.

[AK01] Vikraman Arvind and Johannes Köbler. On pseudorandomness and resource-bounded measure. *Theoretical Computer Science*, 255(1-2):205–221, 2001.

[BDMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM Journal on Computing*, 20(6):1084–1118, 1991.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112. ACM, 1988.

[Blu82] Manuel Blum. Coin flipping by phone. In *24th IEEE Computer Conference (CompCon)*, pages 133–137, 1982.

[BLV06] Boaz Barak, Yehuda Lindell, and Salil Vadhan. Lower bounds for non-black-box zero knowledge. *Journal of Computer and System Sciences*, 72(2):321–391, March 2006.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal of Computing*, 13(4):850–864, 1984.

[BM88] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[BP04] Boaz Barak and Rafael Pass. On the possibility of one-message weak zero-knowledge. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 121–132. Springer, 2004.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 45–64, 2002.

[DDP97] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-efficient non-interactive zero-knowledge (extended abstract). In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, pages 716–726. Springer, 1997.

[DDP99] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Non-interactive zero-knowledge: A low-randomness characterization of $NP$. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 271–280. Springer, 1999.

[DDP02] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Randomness-optimal characterization of two $NP$ proof systems. In *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 179–193. Springer, 2002.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 283–293. ACM, 2000.

[FGM$^+$89] Martin Furer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29:1–28, 1999.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, Maryland, 14–16 May 1990.

[GB01] Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. Available from `http://www.cs.ucsd.edu/users/mihir/papers/gb.html`, August 2001.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM, 1989.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in $NP$ have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.

[Gol01] Oded Goldreich. *Foundations of cryptography : Basic Tools*. Cambridge University Press, Cambridge, 2001.

[Gol04] Oded Goldreich. *Foundations of cryptography II: Basic Applications*. Cambridge University Press, Cambridge, 2004.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. *Advances in Computing Research*, 5:73–90, 1989.

[GST03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3-4):85–130, 2003.

[HILL99] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *30th Annual Symposium on Foundations of Computer Science, 30 October-1 November 1989, Research Triangle Park, North Carolina, USA*, pages 230–235. IEEE, 1989.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if $E$ requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229. ACM, 4–6 May 1997.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 78–95, 2005.

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[Lev87] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[Mil76] Gary L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976.

[MV05] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-01)*, pages 448–457, New York, January 7–9 2001. ACM Press.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[Rab79] Michael Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology, January 1979.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, August 1997.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[Rud97] Steven Rudich. Super-bits, demi-bits, and $\widetilde{\mathrm{NP}}$/qpoly-natural proofs. In *Proceedings of the 1st International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 85–93. Springer, 1997.

[SU05a] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. In *IEEE Conference on Computational Complexity*, pages 212–226. IEEE Computer Society, 2005.

[SU05b] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216 (electronic), 2005.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 80–91. IEEE Computer Society Press, 1982.