Derandomization in Cryptography

Boaz Barak[1,⋆], Shien Jin Ong[2,⋆⋆], and Salil Vadhan[3,⋆⋆⋆]

[1] Weizmann Institute of Science, Rehovot, Israel, boaz@wisdom.weizmann.ac.il.
[2] Massachusetts Institute of Technology, Cambridge, MA, shienjin@mit.edu.
[3] Harvard University, Cambridge, MA, salil@eecs.harvard.edu.

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 $2^{\Omega(n)}$ (Miltersen and Vinodchandran, FOCS '99). 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.

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) [1,2], 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

⋆ Supported by Clore Foundation Fellowship and Israeli Higher Education Committee Fellowship.
⋆⋆ Supported by MIT Eloranta Fellowship and MIT Reed UROP Fund.
⋆⋆⋆ Supported by NSF grants CCR-0205423 and CCR-0133096, and a Sloan Research Fellowship.

D. Boneh (Ed.): CRYPTO 2003, LNCS 2729, pp. 299–315, 2003.

out that the utility of pseudorandom generators was not limited to cryptographic applications. In particular, Yao [2] showed that they could also be used for *derandomization* — efficiently converting randomized algorithms into deterministic algorithms. Pseudorandom generators and their generalization, pseudorandom functions [3], also found a variety of other applications in complexity theory and the theory of computation (e.g., [4,5]).

Focusing on derandomization, Nisan and Wigderson (NW) [6] 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). 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 [7] 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 [8], and has been used to prove complexity-theoretic results which appear unrelated to derandomization [9].

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* [10,11,12,13], 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[1] proof systems, with 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 refutes the intuition that interaction is necessary to achieve secrecy in proof systems. It is obtained by derandomizing the ZAP construction of Dwork and Naor [14].

Result 2 is not the first construction of noninteractive commitment schemes, but is based on assumptions that appear incomparable to previous ones (which

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

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 [15].

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^c. 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 polynomial time, we can obtain the string using an NW-type pseudorandom generator of 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 [10] and Klivans and van Melkebeek [11].[2] Indeed, a sequence of works have constructed such pseudorandom generators under progressively weaker complexity assumptions [10,11,12,13]. Our results make use of the Miltersen–Vinodchandran construction [12] (which gives only a "hitting set generator" rather than a pseudorandom generator, but this suffices for our applications).

Witness Indistinguishable **NP** *Proofs.* In order to make zero-knowledge proofs possible, the seminal paper of Goldwasser, Micali, and Rackoff [17] augmented

[2] 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 [16].

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 [18] 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 [19,20], 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 [18], 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* [21], 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) [21]. Witness indistinguishability suffices for a number of the applications of zero knowledge [21], and also is a very useful intermediate step in the construction of zero-knowledge proofs [22].

Several prior results show that witness-indistinguishable proofs do not require the same degree of interaction as zero-knowledge proofs. Feige and Shamir [21] 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 [14] achieve witness indistinguishability with just 2 messages (assuming trapdoor permutations), whereas this is known to be impossible for zero knowledge [18]. 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 Dwork–Naor ZAPs via an NW-type generator against nondeterministic circuits. 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 on witnesses with respect to R', such that the distributions corresponding to different witnesses 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 [10,

11]. 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 [18] 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 [23,24,25] 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 $O(n^\epsilon + \log(1/s))$ and 2-message witness-indistinguishable proofs (actually, ZAPs) in which the verifier uses only $O(n^\epsilon + \log(1/s))$ random bits, where $\epsilon > 0$ 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 [25] 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 [26] and other cryptographic protocols. Here we focus on perfectly (or statistically) binding and computationally hiding bit commitment schemes. As usual, *noninteractive* bit commitment schemes, in which the commitment phase consists of a single message from the sender to the receiver, are preferred over interactive schemes. There is a simple construction of noninteractive bit commitment schemes from any *one-to-one* one-way function [27,2, 28]. From general one-way functions, the only known construction of bit commitment schemes, namely Naor's protocol [15] (with the pseudorandom generator construction of [29]), 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 semantic 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 required for the NW-type generators we use is a strong one, but it seems to be plausible (see Section 2.4). Perhaps its most sig-

nificant 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 nontrivial 1-message witness-indistinguishable proofs, 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 [27]. 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

2.1 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 1 (Pseudorandom generator). *We say that* $G\colon \{0,1\}^\ell \to \{0,1\}^m$ *is a* (s,ϵ)*-pseudorandom generator against circuits if for all circuits* $C\colon \{0,1\}^m \to \{0,1\}$ *of size at most* s*, it holds that* $|\Pr[C(G(U_\ell)) = 1] - \Pr[C(U_m) = 1]| < \epsilon$*, 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) [1,2] type and Nisan-Wigderson (NW) [6] type generator. Both can be defined for a wide range of parameters, but here we focus on the "classic" settings which we need. A BMY-type generator is the standard kind of pseudorandom generator used in cryptography.

Definition 2 (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 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 [29] 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 3 (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 for all* m*.*[3]

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

We will be interested "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.[4] Impagliazzo and Wigderson [7] 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^ℓ seeds can be enumerated in time poly(m), and hence the generator can be used for complete derandomization. In particular, such a generator implies $\mathbf{BPP} = \mathbf{P}$.

2.2 Nondeterministic Computations and the Class AM

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.

Definition 4. *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.

The class **AM** [30] has two equivalent formulations. The first is as the class of languages with constant-message interactive proofs (see [31] for this definition). 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 (a) when $f(x) = 1$, $\Pr_r[\exists y A(x, r, y) = 1] = 1$ and (b) when $f(x) = 0$, $\Pr_r[\exists y A(x, r, y) = 1] \leq \frac{1}{2}$. Then **AM** is the class of languages decidable by such algorithms $A(x, r, y)$ running in time poly$(|x|)$. The equivalence of the two definitions of **AM** is due to [30,32,33]. More generally, $\mathbf{AMTIME}(t(n))$ denotes the class of languages decidable by probabilistic nondeterministic algorithms running in time $t(n)$, and $\mathbf{i.o.}-\mathbf{AMTIME}(t(n))$ is the class of languages decidable by probabilistic nondeterministic algorithms running in time $t(n)$ for *infinitely many input lengths*.

2.3 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. We say H is *efficient* if its running time is polynomial (in m and s). Hitting set generators are weaker notions of pseudorandom generators.

[4] The running time of the generator is still greater than the size of the circuits it fools.

Definition 5 (Hitting set generators). *We say that H is an ϵ-*hitting set generator *against circuits, if for every circuit $C\colon \{0,1\}^m \to \{0,1\}$ of size at most s, the following holds: If $\Pr[C(U_m) = 1] > \epsilon$, then there exists $y \in H(1^m, 1^s)$ such that $C(y) = 1$.*

One can define analogously hitting set generators against *nondeterministic* and *co-nondeterministic* circuits, and also hitting set generators against nondeterministic and co-nondeterministic *uniform* algorithms. Hitting set generators against co-nondeterministic uniform algorithms will be used only in Section 4.

Note 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). Also note that we allow the hitting set generators to run in greater time than circuits it fools, so they correspond to NW-type generators. Since the error in **AM** proof systems can be made one-sided [33], the existence of an efficient $\frac{1}{2}$-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 [10]. Their construction was based on the assumption that there are languages in **E** that are hard on average for nondeterministic circuits of size $2^{\Omega(n)}$. Klivans and van Melkebeek [11] gave a construction based on a *worst-case* hardness assumption. Their assumption was the existence of languages in **E** with $2^{\Omega(n)}$ worst-case SAT-oracle circuit complexity, that is circuits with SAT-oracle gates. Miltersen and Vinodchandran [12] managed to relax the hardness condition to nondeterministic circuits (yet only obtained a hitting set generator rather than a pseudorandom generator). We state their main result.

Theorem 6 ([12]). [5] *If there exist a function $f \in \mathbf{E}$ such that $S_{\mathrm{N}}(f) = 2^{\Omega(n)}$, then there exist an efficient $\frac{1}{2}$-HSG against co-nondeterministic circuits.*

Shaltiel and Umans [13] subsequently extended Theorem 5 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 [34] extended Theorem 5 to give a hitting set generator against co-nondeterministic *uniform*

[5] [12] presented a $(1-\delta)$-HSG for $\delta = 2^{m^\gamma}/2^m$, but it can be converted into a $\frac{1}{2}$-HSG using dispersers as done implicitly in their paper.

algorithms from *uniform* hardness assumptions. They used the same hitting set generator as Miltersen and Vinodchandran, but proceeded with a better analysis.

Theorem 7 (implicit in [34]). *If* $\mathbf{E} \not\subseteq \mathbf{i.o.}-\mathbf{AMTIME}(2^{\delta n})$ *for some* $\delta > 0$, *then an efficient* $\frac{1}{2}$*-HSG against co-nondeterministic uniform algorithms exists.*

Since nonuniformity can simulate randomness, the existence of a function $f \in \mathbf{E}$ such that $S_{\mathrm{N}}(f) = 2^{\Omega(n)}$ (assumption of Theorem 5) implies that $\mathbf{E} \not\subseteq \mathbf{i.o.}-\mathbf{AMTIME}(2^{\delta n})$ for some $\delta > 0$ (assumption of Theorem 7).

2.4 Discussions

Are the Assumptions Reasonable? Our two results rely on the existence of hitting set generators as constructed in Theorems 5 and 7, which in turn make assumptions about **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 **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 5 and 7 are natural strengthenings of this assumption, where we extend **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 5, then *every* function that is **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 [14] and obtain a *noninteractive witness indistinguishable* (WI) proof system for any language in **NP**. We call this an "**NP** proof system" because it consists of a single message from the prover to the verifier, as is the case in the trivial **NP** proof of simply sending the witness to the verifier.

As in the trivial **NP** proof system, our verifier algorithm will be deterministic. However, our prover algorithm will be *probabilistic*. 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 **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. We define $W(x) = \{w \mid (x,w) \in W\}$. We define $L(W) = \{x \mid \exists w \text{ s.t. } (x,w) \in W\}$. 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 definitions of interactive proofs. Let L be an **NP** language with witness relation W. We say that an interactive proof for L has an *efficient prover* if the prover strategy from the completeness condition can be implemented by an efficient algorithm that when proving that $x \in L$, gets $w \in W(x)$ as an 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 the sense that it consists of only a single message from the prover to the verifier, and that it has a deterministic verifier, and satisfies both perfect completeness and perfect soundness. 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 [21].

Definition 8 (witness indistinguishability, [21]). *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^* and every $x \in L$, and for any $w, w' \in W_L(x)$, the view of V^* when interacting with $P(x,w)$ is computationally indistinguishable from its view when interacting with $P(x,w')$.*

Feige and Shamir also proved that WI is closed under concurrent composition [21].

ZAPs. A *ZAP* [14] is a two-round public-coin interactive proof system that is witness indistinguishable. Dwork and Naor proved the following theorem.

Theorem 9 ([14]). *If trapdoor permutations*[6] *exist, then every language in* **NP** *has a ZAP.*

We note that the construction of ZAPs by [14] is actually based on the possibly 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 possibly weaker assumption.

3.2 Our Result

The main theorem of this section follows.

Theorem 10. *Assume that there exists an efficient* $\frac{1}{2}$*-HSG against co-nondeterministic circuits and that trapdoor permutations exist. Then every language in* **NP** *has a witness-indistinguishable* **NP** *proof system.*

3.3 Proof of Theorem 10

We prove Theorem 10 by converting the ZAPs for languages in **NP** into WI **NP** proofs. 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 π. 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 π such that the transcript (x, r, π) 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 π such that the transcript (x, r, π) is accepting, and 0 otherwise. Note that $\Pr[C_x(U_{\ell(n)}) = 1] > \frac{1}{2}$. Since H is a $\frac{1}{2}$-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 [31][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 [35]. Such a collection is known to exist based on either the RSA or factoring hardness assumptions [36,37].

Protocol 11 (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 π_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, π_i), for every $i \in [1,m]$.*
3. *Accept if the ZAP verifier accepts* all *these transcripts.*

Note that Protocol 11 is indeed a one-message system with a deterministic verifier, that satisfies the perfect completeness property. Thus, to prove Theorem 10, we need to prove that it satisfies both the perfect soundness and the witness indistinguishability property.

Lemma 12. *Protocol 11 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 π_i will make the ZAP verifier accept the transcript (x, r_i, π_i). Therefore, no string $\pi = (\pi_1, \ldots, \pi_m)$ will make the verifier of Protocol 11 accept. □

Lemma 13. *Protocol 11 is a witness indistinguishable (WI) proof system for L.*

Proof. This follows from the fact that witness indistinguishability is preserved under parallel composition. □

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., [31]).

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 [31]. Often, however, *noninteractive* bit commitment schemes are preferred or even crucial. For these, a simpler definition can be given.

Definition 14 (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 $K \in \{0,1\}^{\mathrm{poly}(k)}$, 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 $\{S(0;K)\}_{K\in\{0,1\}^{\mathrm{poly}(k)}}$ and $\{S(1;K)\}_{K\in\{0,1\}^{\mathrm{poly}(k)}}$ are computationally indistinguishable.*

There is a well known construction by Blum [27] of a noninteractive bit commitment scheme based on any *one-to-one* one-way function (using the function's hard-core predicate [2,28]). Naor [15] gave a construction of an *interactive* bit commitment scheme based on any one-way function (using pseudorandom generators [29]).

4.1 Our Result

The main result of this section is the following theorem.

Theorem 15. *Assume that there exists an efficient $\frac{1}{2}$-HSG against co-nondeterministic* uniform *algorithms and that one-way functions exist. Then there exists a noninteractive bit commitment scheme.*

The first condition is true if $\mathbf{E} \not\subseteq \mathbf{i.o.}\text{-}\mathbf{AMTIME}(2^{\Omega(n)})$ (by Theorem 7). We stress that the assumption of efficient $\frac{1}{2}$-HSG against co-nondeterministic *uniform* algorithms is sufficient, even if one wants to obtain a commitment scheme that is secure against *nonuniform* 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.

Our result is incomparable to the previous results on bit commitment schemes. Our assumption is stronger than Naor's [15] (which only requires one-way functions), but we obtain a noninteractive commitment rather than an interactive one. Our assumption seems incomparable to assuming the existence of one-to-one one-way functions.

"Raw" Hardness vs. Hardness with Structure. Note that unlike the assumption of existence of *one-to-one* one-way functions, we do not assume in Theorem 15 that there exist 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 $f \in \mathbf{E} \setminus \mathbf{i.o.}-\mathbf{AMTIME}(2^{\delta n})$. This is because we can instantiate our scheme with a universal one-way-function[7] and a function that is **E**-complete via linear-time reductions such as the function $\mathrm{BH}(\cdot)$ (see discussion in Section 2.4).

4.2 Proof of Theorem 15

Our construction is based on derandomizing Naor's [15] *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 [29]. Naor [15] gave the following protocol for an interactive bit commitment scheme, based on the existence of such a generator.

Protocol 16 (interactive bit commitment scheme [15])
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:

Receiver's step *Select a random* $r \leftarrow \{0,1\}^{3k}$ *and sends r to S.*
Sender's step *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 reveals s and b. R accepts 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 α is distributed according to $G(U_k)$. When the sender commits to 1, α is distributed according to $G(U_k) \oplus r$. For every $r \in \{0,1\}^{3k}$, the distributions $G(U_k)$ and $G(U_k) \oplus r$ are computationally indistinguishable. This implies that Protocol 16 is hiding. Define a string $r \in \{0,1\}^{3k}$ to be *good* for G if for all $s, s' \in \{0,1\}^k$, we have $G(s) \neq G(s') \oplus r$. Naor [15] showed that the probability that a random r in $\{0,1\}^k$ will be good is very high (e.g., at least $1 - 2^{-k}$). If the receiver selected a good r in the first step of the commitment stage of Protocol 16, then there do not exist $s, s' \in \{0,1\}^k$ such that $G(s) = G(s') \oplus r$, so no commitment α can be opened as both a 0 and 1. Since the probability of selecting a good r is high, Protocol 16 is binding.

Our Noninteractive Bit Commitment Scheme. Observe that the only interaction involved in Protocol 16 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

[7] A construction of such a function appears in [38] (cf., [31][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.

we show in the proof of Lemma 18, testing whether r is good can be done by a polynomial-time *co-nondeterministic* uniform algorithm. 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 17 (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* $r_1, \ldots, r_{p(k)} \stackrel{\text{def}}{=} H(1^{3k}, 1^{3k^d})$.
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 17 constitutes a bit commitment scheme (and hence proving Theorem 15), we first observe that the protocol has the hiding property. This means that 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. This fact can be proved using a standard hybrid argument. The next lemma establishes the binding property.

Lemma 18. *Protocol 17 has the binding property.*

Proof. Define the co-nondeterministic 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 α can be opened as both a 0 and 1. Thus, Protocol 17 is perfectly binding. □

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. We say that a function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is a *partially one-to-one one-way function* if the value of x is uniquely determined from $f(x, y)$, yet no probabilistic

polynomial-time algorithm can recover x from $f(x, y)$ (for random $x, y \stackrel{R}{\leftarrow} \{0,1\}^k$) except with negligible probability (in k). It can be shown that partially one-to-one one-way functions exist if and only if noninteractive commitment schemes exist. Thus, a restatement of Theorem 15 is the following.

Corollary 19. *Assume that there exists an efficient $\frac{1}{2}$-HSG against co-nondeterministic uniform algorithms. Then one-way functions imply partially one-to-one one-way functions.*

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

Acknowledgments. We thank the anonymous CRYPTO reviewers for helpful comments.

References

1. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput. **13** (1984) 850–864
2. Yao, A.C.: Theory and applications of trapdoor functions. In: Proc. 23rd FOCS, IEEE (1982) 80–91
3. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. JACM **33** (1986) 792–807
4. Razborov, A.A., Rudich, S.: Natural proofs. JCSS **55** (1997) 24–35
5. Valiant, L.G.: A theory of the learnable. Commun. ACM **27** (1984) 1134–1142
6. Nisan, N., Wigderson, A.: Hardness vs. randomness. JCSS **49** (1994) 149–167
7. Impagliazzo, R., Wigderson, A.: $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In: Proc. 29th STOC, ACM (1997) 220–229
8. Trevisan, L.: Extractors and pseudorandom generators. JACM **48** (2001) 860–879
9. Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: Exponential time vs. probabilistic polynomial time. In: Proc. 16th Conf. on Comp. Complexity, IEEE (2001) 2–12
10. Arvind, V., Köbler, J.: On pseudorandomness and resource-bounded measure. Theoret. Comput. Sci. **255** (2001) 205–221
11. Klivans, A.R., van Melkebeek, D.: Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. SIAM J. Comput. **31** (2002) 1501–1526
12. Miltersen, P.B., Vinodchandran, N.V.: Derandomizing Arthur-Merlin games using hitting sets. In: Proc. 40th FOCS, IEEE (1999) 71–80
13. Shaltiel, R., Umans, C.: Simple extractors for all min-entropies and a new pseudorandom generator. In: Proc. 42nd FOCS, IEEE (2001) 648–657
14. Dwork, C., Naor, M.: Zaps and their applications. In: Proc. 41st FOCS. (2000) 283–293
15. Naor, M.: Bit commitment using pseudorandomness. J. Cryptology **4** (1991) 151–158
16. Rudich, S.: Super-bits, demi-bits, and $\widetilde{\text{NP}}$/qpoly-natural proofs. In: Proc. 1st RANDOM, Springer (1997) 85–93

17. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18** (1989) 186–208
18. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. J. Cryptology **7** (1994) 1–32
19. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: Proc. 20th STOC, ACM (1988) 103–112
20. Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM J. Comput. **20** (1991) 1084–1118
21. Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: Proc. 9th CRYPTO, Springer (1989) 526–545
22. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs under general assumptions. SIAM J. Comput. **29** (1999) 1–28
23. De Santis, A., Di Crescenzo, G., Persiano, G.: Randomness-efficient non-interactive zero-knowledge (extended abstract). In: Proc. 24th ICALP, Springer (1997) 716–726
24. De Santis, A., Di Crescenzo, G., Persiano, G.: Non-interactive zero-knowledge: A low-randomness characterization of NP. In: Proc. 26th ICALP, Springer (1999) 271–280
25. De Santis, A., Di Crescenzo, G., Persiano, G.: Randomness-optimal characterization of two NP proof systems. In: Proc. 6th RANDOM, Springer (2002) 179–193
26. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. JACM **38** (1991) 691–729
27. Blum, M.: Coin flipping by phone. In: 24th IEEE Computer Conference (CompCon). (1982) 133–137
28. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Proc. 21st STOC, ACM (1989) 25–32
29. Hastad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. **28** (1999) 1364–1396
30. Babai, L., Moran, S.: Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes. JCSS **36** (1988) 254–276
31. Goldreich, O.: Foundations of cryptography. Cambridge University Press, Cambridge (2001)
32. Goldwasser, S., Sipser, M.: Private coins versus public coins in interactive proof systems. Advances in Computing Research **5** (1989) 73–90
33. Furer, Goldreich, Mansour, Sipser, Zachos: On completeness and soundness in interactive proof systems. Advances in Computing Research **5** (1989) 429–442
34. Gutreund, D., Shaltiel, R., Ta-Shma, A.: Uniform hardness vs. randomness tradeoffs for Arthur-Merlin games. In: Proc. 18th Conf. on Comp. Complexity, IEEE (2003)
35. Goldreich, O.: Foundations of cryptography : Corrections and additions for volume 1. Available from `http://www.wisdom.weizmann.ac.il/~oded/foc-vol1.html\#err` (2001)
36. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Commun. ACM **21** (1978) 120–126
37. Rabin, M.: Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology (1979)
38. Levin, L.: One-way functions and pseudorandom generators. Combinatorica **7** (1987) 357–363