Are PCPs Inherent in Efficient Arguments?

Guy N. Rothblum*
CSAIL, MIT
Cambridge, MA, USA
Email: rothblum@csail.mit.edu

Salil Vadhan**
SEAS and CRCS, Harvard University
Cambridge, MA, USA
Email: salil@eecs.harvard.edu

***Abstract*—Starting with Kilian (STOC '92), several works have shown how to use probabilistically checkable proofs (PCPs) and cryptographic primitives such as collision-resistant hashing to construct very efficient argument systems (a.k.a. computationally sound proofs), for example with polylogarithmic communication complexity. Ishai et al. (CCC '07) raised the question of whether PCPs are inherent in efficient arguments, and to what extent. We give evidence that they are, by showing how to convert any argument system whose soundness is reducible to the security of some cryptographic primitive into a PCP system whose efficiency is related to that of the argument system and the reduction (under certain complexity assumptions).**

***Keywords*-PCP, MIP, Argument, Black-Box Reduction**

I. Introduction

Probabilistically checkable proofs (PCPs) are one of the greatest successes of the interaction between complexity theory and the foundations of cryptography. The model of PCPs, and the equivalent model of multi-prover interactive proofs, emerged from efforts to find unconditional constructions of zero-knowledge proofs [BGKW] and secure multiparty computation protocols [BGW], [CCD] (replacing the constructions of [GMW1] and [Yao], [GMW2], which relied on computational complexity assumptions). But like their predecessor, interactive proofs, they turned out to be extremely interesting from a purely complexity-theoretic point of view [FRS], particularly through their surprising connection to the inapproximability of optimization problems [FGL$^+$]. The PCP Theorem [AS], [ALM$^+$] is one of the most celebrated results in complexity theory, and has led to a large body of work that continues to generate deep insights.

The PCP Theorem has also provided some returns to cryptography. Specifically, Kilian [Kil] showed how to use PCPs to construct arguments (i.e. computationally sound proof systems) for NP in which the communication complexity is polylogarithmic.[1] Kilian's construction assumes the existence of collision-resistant hash functions with subexponential security. Its zero-knowledge version [Kil] and other variants due to Micali [Mic] and Barak and Goldreich [BG], have found further applications in cryptography [CGH], [Bar]. Moreover, these argument systems provide the asymptotically most efficient approaches for proving general NP statements, and thus are appealing for applications such as proving the correctness of a delegated computation or the safety of a program.

In this paper, we consider the question of whether PCPs are really necessary for very efficient arguments. One of our motivations is simply to better understand the relation between these two fundamental notions in complexity theory and cryptography. In addition, the use of PCPs in efficient argument systems has the drawback that the protocols and their applications inherit the somewhat complex construction and proof of the PCP Theorem. While there have been some substantial advances on simplifying the PCP Theorem [BS], [Din], it remains quite nontrivial and the construction may still be too involved to use in practice.

The question we study here has previously been addressed by Ishai, Kushilevitz and Ostrovsky [IKO]. They showed that by using a stronger cryptographic primitive, namely (additively) homomorphic encryption rather than collision-resistant hashing, it is possible to construct somewhat efficient arguments using the simpler, exponential-length "Hadamard PCP" [ALM$^+$] rather than the polynomial-length PCPs of the full PCP Theorem. Their arguments are only "somewhat efficient" in that they have low (e.g. polylogarithmic) communication from the prover to the verifier, but the verifier-to-prover communication is polynomial (cf. [GVW]).

*Research supported by NSF Grants CCF-0635297, NSF-0729011 and CNS-0430336. Work done in part while visiting U.C. Berkeley and Microsoft Research.

**Work done in part while visiting U.C. Berkeley, supported by the Miller Institute for Basic Research in Science and a Guggenheim Fellowship. Also supported by NSF grant CNS-0831289.

[1] Another important parameter is the computation time of the verifier, but we omit discussion of it in the introduction for the sake of simplicity.

978-0-7695-3717-7/09 $25.00 © 2009 IEEE
DOI 10.1109/CCC.2009.40

Our Results. In this paper, we provide results suggesting that PCPs *are* necessary for constructing efficient arguments. Specifically, we consider a construction of an argument system based on a wide range of cryptographic primitives (e.g. collision-resistant hashing, the RSA assumption, homomorphic encryption), where the computational soundness is based on the security of the primitive via an efficient reduction. That is, there is an algorithm $\mathcal{S}$ such that if $\mathcal{P}^*$ is any prover strategy that convinces the verifier to accept a false statement, then $\mathcal{S}^{\mathcal{P}^*}$ "breaks" the cryptographic primitive.[2] Indeed, we provide a general formulation of a cryptographic primitive and what it means to "break" such a primitive. This formulation is quite general, covering standard primitives such as one-way functions or homomorphic encryption, and specific assumptions such as the hardness of factoring. For such constructions we show how to construct PCPs whose efficiency is related to that of the argument system, the reduction, and a variety of methods for "implementing" the cryptographic primitive (discussed more below).

Informally, our construction works as follows. We view the PCP oracle as specifying a prover strategy $\mathcal{P}_{PCP}$ for the argument system (i.e. the next-message function). The PCP verifier:

1) Chooses an "implementation" C of the cryptographic primitive (to be discussed more below) and sends it to $\mathcal{P}_{PCP}$ (with every query),
2) Runs the verifier of the argument system with $\mathcal{P}_{PCP}$,
3) Runs the reduction $\mathcal{S}$ with $\mathcal{P}_{PCP}$, and
4) Accepts if both the verifier of the argument system accepts (in Step 2) and $\mathcal{S}$ does not break the cryptographic primitive (in Step 3).

To establish soundness, we note that if $\mathcal{P}_{PCP}$ convinces the verifier of the argument system of a false statement, then $\mathcal{S}$ will break the cryptographic primitive; this means that at least one of the acceptance conditions will fail. Thus, soundness of the PCP holds information-theoretically, unconditionally and regardless of the implementation chosen in Step 1 above. For completeness, we need to ensure that the implementation (of the cryptographic primitive) chosen in Step 1 cannot be broken by $\mathcal{S}^{\mathcal{P}}$, where $\mathcal{P}$ is the *honest* prover. We provide several methods for achieving this, some based on complexity assumptions and some unconditional. Below we describe a few of these and the resulting PCP parameters.

[2]Thus, we require that the reduction is "black-box" in its access to $\mathcal{P}^*$, which is true of all of the existing constructions [Kil], [Mic], [BG], [IKO].

Implementation and Parameters. Arguments and PCPs have many parameters, which we treat with as much generality as possible. But for simplicity in the current discussion, we focus on a few parameters of interest, with typical settings. (In particular, we ignore prover and verifier computation time.) Consider an argument system constructed from a cryptographic primitive C for a language $L \in$ NP such that on inputs of length n, the argument system has *prover-to-verifier* communication complexity $\mathrm{polylog}(n)$ and completeness and soundness error $1/3$. Moreover, there is a $\mathrm{poly}(n)$-time reduction $\mathcal{S}$ such that for every $x \notin L$ and every cheating prover $\mathcal{P}^*$, if $\mathcal{P}^*$ convinces the verifier to accept with probability at least $1/3$, then $\mathcal{S}^{\mathcal{P}^*}(C, x)$ "breaks" C with constant probability.

Three key parameters for us are the *verifier-to-prover* communication complexity $v = v(n)$; the number of rounds $r = r(n)$; and the number of queries $\mathcal{S}$ makes to its oracle $q = q(n)$. Kilian's construction of arguments from collision-resistant hash functions (CRHFs) of exponential hardness [Kil] achieves $v = O(\log n + \kappa)$, where κ is the seed length for a CRHF, and $r, q = O(1)$ (here we augment Kilian's construction by basing it on a PCP with constant query complexity, e.g. [AS], [ALM$^+$]). The Ishai et al. [IKO] construction from homomorphic encryption has $v = \mathrm{poly}(n)$ and $r, q = O(1)$.

Given the above, our resulting PCPs for $\mathcal{L}$ will always have constant completeness and soundness errors and alphabet size $\mathrm{polylog}(n)$. The query complexity of our PCPs will be $r+q$, matching the known constructions of arguments from PCPs. The length of our PCPs is $2^{|C|+v}$, where C is the description length of the implementation of the cryptographic primitive generated by the verifier. Note that if $|C| = O(v)$, then this matches known constructions of arguments from PCPs, e.g. exponential-length PCPs correspond to $v = \mathrm{poly}(n)$, and polynomial length PCPs correspond to $v = O(\log n)$.

In this paper we present several approaches for implementing the cryptographic primitive used by the construction. Recall that our PCP verifier needs to generate an implementation C of the cryptographic primitive that cannot be broken by $\mathcal{S}^{\mathcal{P}}$, where $\mathcal{P}$ is the *honest* prover of the argument system. This is done in a variety of different ways, we outline a few below:

- The general framework above is formalized in Section IV, where we also present a natural (and efficient) instantiation. Assume that we have a secure implementation of the cryptographic primitive in the usual sense, e.g. that collision-resistant hash functions exist or that homomorphic encryption schemes exist, with whatever security parameter is used by the underlying argument system (typically $\mathrm{polylog}(n)$ to achieve polylogarithmic communication) and security against $\mathrm{poly}(n)$-time adversaries. Such a primitive cannot be broken by $\mathcal{S}^{\mathcal{P}}$, because this is a $\mathrm{poly}(n)$-time algorithm. Then, since the implementation C is described by a fixed algorithm (which gets a security parameter as input), it can be hardwired into the PCP protocol. Here we obtain a standard (information-theoretically sound) PCP, where completeness relies on the assumption that the implementation of the primitive is secure. We view the assumption we use here as quite natural, as if there were no secure implementation of the primitive, then the original construction was a significantly less interesting object to begin with.

- In Section V, we consider more restrictive notions of reductions, where all entities in the argument system are given only black-box access to a cryptographic primitive such as a one-way function, pseudorandom function family, or collision resistant hash family. We show that, in some cases, we can *minimize* or *remove altogether* the computational assumptions made in the results presented above (for such fully black-box reductions). To do so, we observe that the implementation of the cryptographic primitive we use need only be secure against $\mathcal{S}^{\mathcal{P}}$: a single fixed polynomial-time algorithm. Obtaining an implementation that is secure against a fixed polynomial time bounded algorithm can be considerably easier than obtaining an implementation secure against *any* polynomial-time algorithm (the usual requirement from cryptographic primitives). For example, we can actually construct such pseudorandom functions C with seed length $|C| = O(\log n)$ using [IW], under the worst-case complexity assumption that $E = DTIME(2^{O(n)})$ does not have circuits of size $2^{o(n)}$.
We can also obtain *unconditional* implementations of collision-resistant hash functions against $\mathcal{S}^{\mathcal{P}}$ using a family of $\mathrm{poly}(n)$-wise independent hash functions, yielding $|C| = \mathrm{poly}(n)$. Alternatively, by picking $\mathrm{poly}(n)$ hash functions at random from such a family and hardwiring them into the PCP verifier and prover, we can obtain a *nonuniform* PCP construction with $|C| = O(\log n)$. (This can also be viewed as a uniform construction in the common reference string model.)[3]

We emphasize that even though we use complexity assumptions in some of our transformations, the resulting PCPs achieve the standard, statistical definition of soundness — there does not exist *any* proof oracle that can convince the verifier to accept a false statement, except with small probability. Indeed, the complexity assumptions are only used for the *completeness* of (some of) our constructions, in order to ensure that the honest proof oracle that describes the prover does not inadvertently allow (the reduction $\mathcal{S}$) to break the primitive. This conditional completeness differs from the soundness of the original argument system, which also held under the same assumption, but was only guaranteed against bounded malicious provers.

Perspective. Like all negative results regarding reductions, our results do not entirely preclude the possibility of obtaining efficient arguments "without PCPs," and may be alternatively interpreted as suggesting avenues for doing so. One possibility is to make non-black-box use of the cheating prover strategy $\mathcal{P}^*$ in the reduction from breaking the cryptographic primitive to violating soundness (or at least to use the fact that $\mathcal{P}^*$ is efficient). Another is to make use of reductions $\mathcal{S}$ that make many queries q to the cheating prover, even when soundness is violated with constant probability. (If $q = \mathrm{poly}(n)$, we get a PCP with $\mathrm{poly}(n)$ queries, which is trivial.) Another direction is to use a reduction where a malicious prover that breaks soundness with even say constant probability only breaks the cryptographic primitive used with polynomially small advantage. Such a reduction would also yield only a PCP with polynomial query complexity. Known reductions are not of this type.

Organization. Some of the proofs and results, as well as an overview of known constructions of arguments and their parameters, were omitted form this extended abstract. See the full version for further details.

II. Preliminaries and Definitions

Let $[n]$ be the set $\{1, 2, \dots n\}$. For $x, y \in \{0,1\}^n$ we use $x \circ y$ to denote the concatenation of x and y (a string in $\{0,1\}^{2n}$). For a (discrete) distribution D over a set X we denote by $x \sim D$ the experiment of selecting $x \in X$ by the distribution D. A function $f(n)$ is *negligible* if it is smaller than any (inverse) polynomial. We refer the reader to [Gol1], [Gol2] for complete definitions of standard cryptographic objects used in this work such as one-way functions, distribution ensembles, collision

[3]Using the connection between PCPs and hardness of approximation, this construction can also be viewed as a randomized reduction from $\mathcal{L}$ to an approximate version of $MAXSAT$ (cf., [BGS]).

resistant hash functions and pseudo-random functions. We emphasize that throughout this work whenever we make or refer to hardness assumptions, the assumptions are always against non-uniform adversaries.

We present definitions of the two types of proof system we consider in this work

Definition II.1 (Argument System $(\mathcal{P}, \mathcal{V})$ [BCC], [GMR]). An argument system for a language $\mathcal{L} \in \mathcal{NTIME}(g(n))$ consists of two interactive machines. $\mathcal{P}(x, w)$ gets an input x and advice $w \in \{0,1\}^{g(|x|)}$ (usually an NP witness to the input's membership in $\mathcal{L}$). $\mathcal{V}(x)$ gets the input x. The requirements are:

- Completeness $c(n)$. For every $x \in \mathcal{L}$ and corresponding witness w, the interaction of $\mathcal{V}(x)$ with $\mathcal{P}(x, w)$ (sometimes denoted $(\mathcal{P}(x, w), \mathcal{V}(x))$) makes $\mathcal{V}$ accept with probability at least $c(n)$
- Soundness $s(n)$ vs. size $f(n)$. For every $x \notin \mathcal{L}$ and every cheating $\mathcal{P}^*$ of (non-uniform) size at most $f(n)$, the probability that $(\mathcal{P}^*(x), \mathcal{V}(x))$ makes $\mathcal{V}$ accept is at most $s(n)$.

There are many complexity measures of an argument system that will interest us, such as the communication complexity (in each direction), the round complexity, the size of the honest prover and verifier, and more.

Definition II.2 (PCP, Probabilistically Checkable Proof $(\mathcal{P}, \mathcal{V})$). An argument system for a language $\mathcal{L} \in \mathcal{NTIME}(g(n))$ consists of a non-adaptive (i.e. stateless) machine $\mathcal{P}$ and an oracle machine $\mathcal{V}$. $\mathcal{P}(x, w)$ gets an input x, advice $w \in \{0,1\}^{g(|x|)}$ (usually an NP witness to the input's membership in $\mathcal{L}$) and an input oracle query. $\mathcal{V}(x)$ gets the input x. The requirements are:

- Completeness $c(n)$ For every $x \in \mathcal{L}$ and corresponding witness w, the probability that $\mathcal{V}(x)$ accepts when it is run with $\mathcal{P}(x, w)$ as its oracle (we denote this as $\mathcal{V}(x)^{\mathcal{P}(x,w)}$), is at least $c(n)$.
- Soundness $s(n)$ For every $x \notin \mathcal{L}$ and every non-adaptive cheating $\mathcal{P}^*$ oracle, the probability that $\mathcal{V}(x)^{\mathcal{P}^*(x)}$ accepts is at most $s(n)$.

There are many complexity measures of an PCP system that have been extensively studied. In this work we focus on the query complexity (the number of oracle calls $\mathcal{V}$ makes), the alphabet size (the size of $\mathcal{P}$'s output), the PCP length (the number of possible $\mathcal{P}$ input queries), the size of the honest prover and verifier, and more.

III. Cryptographic Primitives and Reductions

In this section we consider reductions from cryptographic primitives to computationally sound argument systems. We would like to consider a general notion of a cryptographic primitive and of a reduction. The notion we present here already captures a host of cryptographic primitives such as one-way functions, encryption schemes, specific assumptions, and more. See the discussion below regarding how they fit the formalism.

Definition III.1 (Cryptographic Primitive). A *cryptographic primitive* $(\mathcal{C}, \mathcal{T})$ is defined by a class $\mathcal{C}$ of circuits and a *testing* procedure $\mathcal{T}$. For a candidate C in $\mathcal{C}$ (a circuit in the class), we say that an interactive adversary $\mathcal{A}$ (κ, ε)*-breaks* C if:

$$\Pr_{\mathcal{T}\text{'s coins}}\left[\mathcal{T}^{\mathcal{A}}(C, 1^{\kappa}, 1^{\lceil 1/\varepsilon \rceil}) \text{ accepts }\right] \geq 2/3$$

On the other hand, we say C *is* (κ, ε)*-secure* against $\mathcal{A}$ if:

$$\Pr_{\mathcal{T}\text{'s coins}}\left[\mathcal{T}^{\mathcal{A}}(C, 1^{\kappa}, 1^{\lceil 1/\varepsilon \rceil}) \text{ accepts }\right] \leq 1/3$$

where in both cases $\mathcal{T}$ is given access to the circuit C. Throughout this work we deal only with C and $\mathcal{A}$ for which there is a *promise* that either $\mathcal{A}$ breaks C, or C is secure against $\mathcal{A}$. The input parameter κ is typically used to denote a security parameter that bounds the input and output sizes of circuit C (circuits that don't meet this bound make $\mathcal{T}(C, 1^{\kappa}, \cdot)$ accept immediately).[4] Intuitively, the parameter ε is used to specify a "threshold" for the success probability of $\mathcal{A}$ in breaking the primitive, see the examples below.

Note that the above notion can be extended to consider classes of circuit distributions (rather than circuits, or circuit distributions with support size 1, as done above). For simplicity and clarity we use the more restricted notion (Definition III.1 suffices to capture all the cryptographic primitives we consider in this work). We proceed by considering several examples and how they fit into the above definition of a cryptographic primitive:

1) One-Way Functions. Here $\mathcal{C}$ is the class of circuits computing a function, say from $\{0,1\}^{\kappa}$ to $\{0,1\}^{\kappa}$. Given a circuit C and adversary $\mathcal{A}$, the tester $\mathcal{T}^{\mathcal{A}}(C, 1^{\kappa}, 1^{\lceil 1/\varepsilon \rceil})$ chooses $O(1/\varepsilon)$ random inputs to the function, applies C to each of the inputs, and runs $\mathcal{A}$ on each of the outputs. $\mathcal{T}$ accepts if the adversary inverts C on at least one of these outputs (i.e. $C(A(C(x))) = C(x)$ for one of the inputs x). If $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is a (length-preserving) one-way function, this

[4]Note that κ could also be used to bound the size of the circuit C, we will not do so in this work.

means that it is computable in polynomial time (in its input length), and for every PPT $\mathcal{A}$ and polynomial $p(\cdot)$, for sufficiently large κ, f_κ is $(\kappa, 1/p(\kappa))$-secure against $\mathcal{A}_\kappa$. I.e., it holds that: $\Pr[\mathcal{T}^{\mathcal{A}_\kappa}(f_\kappa, 1^\kappa, 1^{p(\kappa)})$ accepts $] \leq 1/3$, where f_κ and $\mathcal{A}_\kappa$ are the restrictions of f and $\mathcal{A}$ to inputs of length κ.

We can also consider subexponentially-hard one-way functions. If $f : \{0,1\}^* \to \{0,1\}^*$ is a (length-preserving) subexponentially-hard one-way function, then it is computable in polynomial time (in its input length), and for some constant $\delta > 0$ and every probabilistic algorithm $\mathcal{A}$ running in time at most 2^{κ^δ}, for sufficiently large κ, f_κ is $(\kappa, 1/2^{\kappa^\delta})$-secure against $\mathcal{A}_\kappa$.

2) Collision-Resistant Hash Families. Here $\mathcal{C}$ is the class of circuits that evaluate families of shrinking hash functions say from $\{0,1\}^{2\kappa}$ to $\{0,1\}^\kappa$. I.e., C in $\mathcal{C}$ gets as input a seed s and an input x and outputs the function $C_s(x) = C(s,x)$. The tester $\mathcal{T}$ chooses $O(1/\varepsilon)$ random seeds $\{s_1, s_2, \ldots s_{O(1/\varepsilon)}\}$, and asks $\mathcal{A}$ to find a collision on each of them, it accepts if $\mathcal{A}$ succeeds on at least one (i.e. if given for any of the seeds s_i, the adversary $\mathcal{A}(s_i)$ finds x and x' such that $C(s_i, x) = C(s_i, x')$).
3) Hardness of Factoring. We can also view *specific* number-theoretic (or other) assumptions as cryptographic primitives in our framework. To capture, for example, the assumption that factoring is hard, we have a single circuit C_κ for every value of the security parameter. This circuit C_κ is the canonical circuit that pick two random $\kappa/2$-bit primes and outputs their product. The tester $\mathcal{T}$ takes $O(1/\varepsilon)$ random samples (numbers) $\{n_1, n_2, \ldots, n_{O(1/\varepsilon)}\}$ from the distribution. It then asks $\mathcal{A}$ to factor each of these numbers, and accepts if $\mathcal{A}$ succeeds on at least one ($\mathcal{A}$ finds a non-trivial factorization of some n_i into two prime factors).
4) Homomorphic Encryption Scheme. A homomorphic encryption scheme is a (public or secret key) scheme with a special homomorphic evaluation procedure that can be used on a sequence of ciphertexts to compute an encryption of some function f of the plaintexts (common functions include addition and multiplication). The scheme remains semantically secure against an adversary who is given a circuit computing this homomorphic evaluation procedure. See [Gol2] for more details on semantically secure encryption schemes.

In this example, $\mathcal{C}$ is the class of circuits that perform key generation, encryption, decryption and homomorphic evaluation procedures. The tester uses $C \in \mathcal{C}$ to generate a key and feeds the adversary with this key and with the homomorphic evaluation procedure (for public-key schemes, the adversary is also given the encryption procedure). The tester and adversary then run the semantic security game $O(1/\varepsilon^2)$ times, and the tester accepts if the adversary has advantage ε in breaking the scheme's semantic security in these experiments.

Discussion. Note that the definition of a cryptographic primitive is decoupled from the question of whether there exists an *implementation* of the primitive that is (κ, ε)-secure against a collection of adversaries. We note also that the related work of Naor [Nao] considers general notions of cryptographic assumptions and primitives. There, however, the primary focus is classifying cryptographic assumptions according to how efficiently they can be *falsified*. In that setting, one of the goals is designing specific procedures that not only break the cryptographic assumption (assuming that it can be broken), but that do so in a way that can be verified very efficiently. We, on the other hand, focus on verifying that an *arbitrary* adversary (provided by a security reduction) breaks the cryptographic primitive. Nonetheless, falsifiable (and even only somewhat falsifiable, cf. [Nao]) assumptions naturally fall into our framework of a cryptographic primitive. Non-falsifiable assumptions, such as the knowledge of exponent assumption [Dam], may not fit into our notion. This is because there is no "testing procedure" that can be used to tell whether an adversary breaks the assumption.

Now that we have presented our notion of a cryptographic primitive, we proceed to define a *reduction* from a cryptographic primitive to an argument system.

Definition III.2 (Reduction)**.** A reduction $\mathcal{R} = (\mathcal{P}, \mathcal{V}, \mathcal{S}, (\mathcal{C}, \mathcal{T}), \kappa(\cdot), \varepsilon(\cdot))$ from a cryptographic primitive defined by $(\mathcal{C}, \mathcal{T})$ to an argument system for a language $\mathcal{L}$ consists of several components. We use x to denote an n bit input whose membership is being proved, and w to denote the prover's auxiliary input (usually the NP witness).

1) A cryptographic primitive $(\mathcal{C}, \mathcal{T})$ as in Definition III.1.
2) Two functions $\kappa : \mathbb{N} \to \mathbb{N}$, $\varepsilon : \mathbb{N} \to [0,1]$ that determine the parameters of the cryptographic primitive as a function of the input length. The function $\kappa(\cdot)$ determines the security parameter,

and $\varepsilon(\cdot)$ determines the advantage of an adversary who breaks the argument's soundness in breaking the cryptographic primitive.[5]

3) Two interactive oracle machines: a prover $\mathcal{P}(C, x, w)$ and verifier $\mathcal{V}(C, x)$ with access to a candidate circuit C in $\mathcal{C}$.
4) A proof of security: an oracle machine $\mathcal{S}$ with black-box access to a cheating prover $\mathcal{P}^*(C, x)$ that gets as input C in $\mathcal{C}$ and $x \in \{0,1\}^n$.

We require that $(\mathcal{P}(C, \cdot, \cdot), \mathcal{V}(C, \cdot))$ is complete for *every* candidate $C \in \mathcal{C}$. For security, we require that if a cheating prover $\mathcal{P}^*(C, x)$ violates soundness for some $x \notin \mathcal{L}$ and C, then $\mathcal{S}^{\mathcal{P}^*}(\cdot, x)$ breaks the (supposedly hard) C. If C is indeed hard to break, then the argument system is thus sound. We state these requirements formally below:

1) Completeness $c(n)$. For every C in $\mathcal{C}$, given $x \in \mathcal{L}$ and a valid witness w, the prover $\mathcal{P}(C, x, w)$ convinces $\mathcal{V}(C, x)$ with probability at least $c(n)$.
2) Security proof of soundness $s(n)$. For every C in $\mathcal{C}$, every n-bit input $x \notin \mathcal{L}$ and every cheating prover $\mathcal{P}^*(C, x)$: if $(\mathcal{P}^*(C), \mathcal{V}(C))(x)$ accepts with probability at least $s(n)$, then $\mathcal{S}^{\mathcal{P}^*(\cdot,x)}$ breaks C, i.e.:

$$\Pr\left[\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*(\cdot,x)}}(C, 1^{\kappa(n)}, 1^{\lceil 1/\varepsilon(n) \rceil}) \text{ accepts}\right] \geq 2/3$$

For simplicity, one can think of $\varepsilon(n) = s(n)^{O(1)}$ throughout this work.

We assume throughout that $s(n) \leq 0.1$ and $c(n) \geq 0.9$. We use $t(n)$ to denote the circuit size $\mathcal{S}^{\mathcal{P}}$ (i.e., $t(n)$ is $|\mathcal{S}| \cdot |\mathcal{P}|$, here we refer only to the honest $\mathcal{P}$), and $q(n)$ to denote the number of $\mathcal{P}^*$-oracle queries made by $\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*}}$.[6] We use $v(n)$ to denote a bound on the number of bits sent from $\mathcal{V}$ to $\mathcal{P}$, $u(n)$ to denote a bound on the length (in bits) of each of $\mathcal{P}$'s answers, and $r(n)$ to denote the number of rounds of communication of $(\mathcal{P}, \mathcal{V})$.

In the reduction notion of Definition V.1, all the algorithms in the argument system (prover, verifier, tester $\mathcal{T}$) get access to C's explicit representation. The only "black-box" access in the definition is the security proof's access to the cheating prover. This is quite a general notion of reduction. See Reingold, Trevisan and Vadhan [RTV] for a discussion of different notions of reductions. In this work we also consider more restricted notions. Black-box reductions are reductions where the algorithms access C as a black box. See Section V for a discussion and definitions of these more restricted types of reductions.

[5]We find it convenient to have the reduction determine the security parameter $\kappa = \kappa(n)$ and the advantage in breaking the cryptographic primitive $\varepsilon = \varepsilon(n)$, rather than give κ as input to all the algorithms.

[6]Throughout, whenever we refer to a bound on a parameter that depends on P^* we mean the worst case bound over the input, the cheating prover, etc. for input length n. Note that these bounds may also depend on the security parameter $\kappa(n)$, which is a parameter of the reduction.

IV. From Arguments to PCPs

In this section, we take any reduction $\mathcal{R} = (\mathcal{P}, \mathcal{V}, \mathcal{S}, (\mathcal{C}, \mathcal{T}), \kappa(\cdot), \varepsilon(\cdot))$ from a cryptographic primitive specified by $(\mathcal{C}, \mathcal{T})$ to an argument system for a language $\mathcal{L}$, and construct from it a PCP for $\mathcal{L}$.

A. A Generic Transformation

For all of our results, we need an additional property from the reduction. We require that it is possible to generate candidates C in $\mathcal{C}$ for the cryptographic primitive, that cannot be broken by the security proof $\mathcal{S}^{\mathcal{P}}$ when it runs with the *honest* prover (except with small probability). We formalize this property below.

Property IV.1. *The reduction $\mathcal{R}$ (with soundness $s(n)$) has a (polynomial time deterministic) generation procedure $\mathcal{G}(1^n)$ that outputs a candidate C in $\mathcal{C}$ such that for every $x \in \{0,1\}^n$ and advice string $w \in \{0,1\}^{\mathrm{poly}(n)}$ given to the prover $\mathcal{P}$, C is $(\kappa(n), \varepsilon(n))$-secure against $\mathcal{S}^{\mathcal{P}(\cdot,x,w)}(\cdot, x)$.*[7]

In Section V we extend this notion to probabilistic generators $\mathcal{G}$. We will restrict our attention to deterministic $\mathcal{G}$ throughout this section.

We now specify a "generic" PCP construction for reductions with Property IV.1. We will later show how to instantiate this generic construction for specific cryptographic primitives, by constructing a generator $\mathcal{G}$ that meets Property IV.1 (unconditionally or under various assumptions). We view the verifier for the PCP as an oracle machine (with oracle access to the proof or oracle-prover). We run the generator $\mathcal{G}$ to generate a candidate C. The generic verifier $\mathcal{V}_{PCP}$ and the (honest) prover oracle $\mathcal{P}_{PCP}$ depend on this candidate C. The verifier and prover are specified in Figures 1 and 2.

The intuition is that if for $x \notin \mathcal{L}$ a cheating PCP prover $\mathcal{P}^*_{PCP}$ makes the verifier $\mathcal{V}_{PCP}$ accept with probability $s(n)$ or greater in Step 1, then the reduction $\mathcal{R}$ guarantees that in Step 2, the security proof $\mathcal{S}^{\mathcal{P}^*_{PCP}}$ will break C correctly with advantage $\varepsilon(n)$ (and $\mathcal{V}_{PCP}$ rejects). This guarantees soundness. On the other hand, when $x \in \mathcal{L}$, we know by Property IV.1 that C is $(\kappa(n), \varepsilon(n))$-secure against the security proof run with

[7]Recall that in Definition III.1 we captured "security against $\mathcal{A}$" by saying that the testing procedure accepts $\mathcal{A}$ with probability at most 1/3.

Verifier $\mathcal{V}_{PCP}(C, x)$

1) Choose random coins for $\mathcal{V}$. Simulate $\mathcal{V}(C, x)$ in the interactive argument system using these coins and the candidate C, using the PCP prover $\mathcal{P}_{PCP}$ to obtain the messages of the argument system's prover $\mathcal{P}(C, x, w)$. Thus, each query to $\mathcal{P}_{PCP}$ specifies a transcript of the interactive argument (The verifier's messages are computed using the existing transcript and the random coins chosen.) If $\mathcal{V}$ rejects, then reject. Otherwise, continue to Step 2.
2) Repeat the following $O(\log(1/\alpha))$ times, where $\alpha = \alpha(n)$ is a parameter:
   Run the tester $\mathcal{T}^{\mathcal{S}^{\mathcal{P}_{PCP}}}(C, 1^n, 1^{\lceil 1/\varepsilon(n) \rceil})$ with independent random coins to check whether $\mathcal{S}^{\mathcal{P}_{PCP}}$ breaks C. Here $\mathcal{P}_{PCP}$ plays the role of answering $\mathcal{S}$'s oracle queries to P^*. Again, each query to $\mathcal{P}_{PCP}$ specifies a transcript of the interactive argument.
   If in at least half of these iterations $\mathcal{T}$ accepts, then reject. Otherwise accept.

Figure 1. Verifier $\mathcal{V}_{PCP}$

(Honest) PCP Proof Oracle $\mathcal{P}_{PCP}(C, x, w)$

For any query specifying a transcript of past messages for the interactive argument, simulate $\mathcal{P}(C, x, w)$ on this transcript and output its next message.

Figure 2. (Honest) PCP Proof Oracle $\mathcal{P}_{PCP}$

the honest prover (and so the verifier should usually accept). This guarantees completeness. We formalize this in the theorem below.

Theorem IV.2. *Let $\mathcal{R} = (\mathcal{P}, \mathcal{V}, \mathcal{S}, (\mathcal{C}, \mathcal{T}), \kappa(\cdot), \varepsilon(\cdot))$ be a reduction from a cryptographic primitive specified by $(\mathcal{C}, \mathcal{T})$ to an argument system for a language $\mathcal{L}$ as in Definition III.2. Suppose furthermore that $\mathcal{R}$ satisfies Property IV.1 and has a generator $\mathcal{G}$ for hard candidates.*

Let $c = c(n)$ and $s = s(n)$ be the completeness and soundness of the argument system, and take $\varepsilon = \varepsilon(n)$ and $\kappa = \kappa(n)$. Recall that $v = v(n)$ bounds the communication from $\mathcal{V}$ to $\mathcal{P}$, the value $u = u(n)$ bounds $\mathcal{P}$'s answer lengths, the value $r = r(n)$ bounds the number of rounds, and $q = q(n)$ bounds the number of $\mathcal{P}^*$-queries made by $\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*}}(C, 1^n, 1^{\lceil 1/\varepsilon \rceil})$.

Then $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ is a PCP for $\mathcal{L}$ with completeness $c - \alpha$ and soundness $\max\{s, \alpha\}$. The number of queries is $r + O[\log(1/\alpha) \cdot q]$. The alphabet size is 2^u. The length of the PCP is 2^v. Furthermore, the PCP oracle can be constructed in time polynomial in that of $\mathcal{P}(C, x, w)$. The running time of the PCP verifier is polynomial in that of $\mathcal{G}$, of $\mathcal{V}(C, x)$ and of $\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*}}(C, 1^n, 1^{\lceil 1/\varepsilon \rceil})$.

Proof: We begin by analyzing the proposed construction's alphabet size, length and query number:

- Query number: The verifier $\mathcal{V}_{PCP}$ makes r queries to $\mathcal{P}_{PCP}$ in Step 1 (one for each round of communication between $\mathcal{V}$ and $\mathcal{P}$). It then runs $O(\log(1/\alpha))$ simulations of $\mathcal{S}$, each of which makes q queries. The total number of queries is thus $r + O[(\log(1/\alpha)) \cdot q]$.
- Alphabet size: The answers of $\mathcal{P}_{PCP}$ are messages sent by the prover $\mathcal{P}$ in the interactive argument, their length is bounded by u and the alphabet size is bounded by 2^u.
- PCP length: Each query made by $\mathcal{V}_{PCP}$ includes a transcript for the interactive argument . The length of each such query is thus v, and the length of the PCP is 2^v.

We now turn our attention to completeness and soundness. For soundness, suppose $x \notin L$ but $\mathcal{P}^*_{PCP}$ makes the verifier $\mathcal{V}_{PCP}$ accept in Step 1 with probability at least s. We view $\mathcal{P}^*_{PCP}$ as a cheating prover P^* for the interactive argument. By the properties of the reduction $\mathcal{R}$, we know that $\mathcal{S}^{P^*}(\cdot, x)$ will have advantage ε in breaking C. This means that in Step 2, every time that $\mathcal{V}_{PCP}$ simulates $\mathcal{T}$, it accepts with probability at least 2/3. Thus (repeating $\Theta(\log(1/\alpha))$ times), the verifier $\mathcal{V}_{PCP}$ will reject in Step 2 with all but probability α. If the probability of "accepting" (i.e. not rejecting) in Step 1 is greater than s, then the verifier accepts in Step 2 with probability at most α. Hence, the total probability of accepting is at most $\max\{s, \alpha\}$.

For completeness, in Step 1 of $\mathcal{V}_{PCP}$'s operation, when it runs the argument system's $\mathcal{V}$, it will accept with probability at least c by the completeness of $(\mathcal{P}, \mathcal{V})$. By Property IV.1, the candidate C is (κ, ε)-secure against $\mathcal{S}^{\mathcal{P}_{PCP}}$. So in every iteration of Step 2, $\mathcal{T}$ will reject with probability at least 2/3. Repeating $\Omega(\log(1/\alpha))$ times, the probability that the verifier rejects in Step 2 is at most α. Taking a union bound, the total probability of accepting when $x \in L$ and the prover is honest is at least $c - \alpha$. ∎

B. Constructions Under Cryptographic Assumptions

As an immediate corollary of Theorem IV.2, we obtain conditional constructions of PCPs from argument systems. If there is indeed a computationally hard candidate for the cryptographic primitive on which the argument's construction is based, then this candidate immediately satisfies Property IV.1. We view this as a natural assumption to make: presumably we consider the construction of an argument to be meaningful because we believe that the cryptographic primitive has a secure implementation. Given such an implementation, we get a PCP (with statistical soundness) "for free". We can use the candidate to construct the PCP. In fact, it suffices that the implementation is secure against (the reduction run with) the *fixed polynomial-time bounded honest prover*, so we can even make do with a cryptographic primitive that is only secure against this fixed algorithm. We emphasize that *the soundness of the PCP obtained is unconditional and information-theoretic*; it is only completeness that is based on the cryptographic assumption. In fact, we only need hardness against the security proof when it uses the *honest* prover (we elaborate and build on this in subsequent sections).

Here the notion of reduction from arguments to cryptographic primitives used is the general notion of Definition III.2, i.e. the reduction is only black-box only in the adversary. In particular, we obtain the following (informal) corollary:

Corollary IV.3 (Informal)**.** *Let $\mathcal{R}$ be a reduction from a cryptographic primitive to a computationally sound argument system for language $\mathcal{L}$. If there exists a secure implementation of the cryptographic primitive, then the argument system can be used to construct a PCP as in Theorem IV.2.*

In particular, if there exists a family of collision-resistant hash functions, then any reduction from CRHF to a computationally sound argument system can be used to construct a PCP. If there exists an additively homomorphic encryption scheme, then any reduction from additively homomorphic encryption to a computationally sound argument system can be used to construct a PCP.

Perspective from known constructions. We first examine the known reductions using collision-resistant hashing for NP arguments [Kil], [Mic], [BG]. Taking κ to be the security parameter, the communication from $\mathcal{V}$ to $\mathcal{P}$ is $v(n) = O(\log(n) + \kappa)$ (specifying the hash and $O(1)$ PCP queries), the length of prover answers is $u(n) = O(\log(n) \cdot \kappa)$, and the number of rounds is $r(n) = O(1)$. The number of calls $\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*}}$ makes to $\mathcal{P}^*$ is $q(n) = O(1)$ (for constant soundness and advantage in breaking the primitive). Theorem IV.2 gives (for any instantiation) a PCP with constant completeness and soundness, $O(1)$ queries, alphabet size $2^{O(\log(n)\cdot\kappa)}$, and proof length $\mathrm{poly}(n) \cdot 2^{\kappa}$. Thus, if we take a polylogarithmic security parameter, the PCP length is quasipolynomial. This does not quite match the Kilian [Kil] construction (which needed a polynomial-length PCP), but as we show in Section V, we can actually (under complexity assumptions) get implementations of the CRHF that suffice for the construction above and with logarithmic κ. This yields a polynomial-length PCP from any (black-box) construction with the parameters of [Kil].

If we examine the reduction of [IKO], there the communication from the verifier to the prover is κ times the logarithm of the length of the PCP being used, $v(n) = \mathrm{poly}(n) \cdot \kappa$ (in their case the PCP used was exponential, and so $v(n)$ is polynomial). The communication from the prover to the verifier is $u(n) = O(\kappa)$, and the number of rounds is $r(n) = O(1)$. Again, the number of calls $\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*}}$ makes to $\mathcal{P}^*$ is $q(n) = O(1)$ (for constant soundness and advantage in breaking the encryption). Theorem IV.2 gives (for any instantiation) a PCP with constant completeness and soundness, $O(1)$ queries, alphabet size $2^{O(\kappa)}$, and proof length $2^{\mathrm{poly}(n)\cdot\kappa}$ (as should be expected, because they started with an exponential length PCP).

V. Weakening or Eliminating Computational Assumptions

In this section we consider more restricted reductions that those of Section IV-B, and obtain PCP constructions with better parameters. The main idea will be to build an implementation for the cryptographic primitive used by the reduction that is only secure against one specific adversary: the adversary which runs the reduction together with the *honest* argument prover (such an implementation still suffices for arguing completeness a la Theorem IV.2). In this section, we will look at *(fully) black-box reductions*. Proofs have been omitted for lack of space. See the full version for proofs and for definitions and results about *black-box reductions with bounded adaptivity*.

Definition V.1 (Black-Box Reduction.)**.** A reduction $\mathcal{R} = (\mathcal{P}, \mathcal{V}, \mathcal{S}, (\mathcal{C}, \mathcal{T}), \kappa(\cdot), \varepsilon(\cdot))$ is a *(fully) black-box* reduction if it is a reduction, as in Definition III.2, and also $\mathcal{P}$ and $\mathcal{S}$ only have black-box access to C, i.e. they access C as an oracle. Here $t(n)$ also bounds the number of oracle calls made by $\mathcal{S}^{\mathcal{P}}$ (as $t(n)$ is the total combined size of this procedure).

Probabilistic Candidate Generator. To get unconditional results and results under weaker (worst-case) assumptions, we need to generalize Property IV.1. We need to extend that property to the case where we do not have a deterministic generator that outputs a single hard implementation, but rather a probabilistic generator outputs a hard implementation (for a specific algorithm) w.h.p.

Property V.2. *The reduction $\mathcal{R}$ (with soundness $s(n)$) has a* probabilistic *polynomial-time generation procedure $\mathcal{G}(1^n)$ that outputs a candidate C in $\mathcal{C}$ such that for every $x \in \{0,1\}^n$ and advice string $w \in \{0,1\}^{\mathrm{poly}(n)}$ given to the prover $\mathcal{P}$:*

$$\Pr_{\mathcal{G}\text{'s coins}}\left[C \text{ is } (\kappa(n), \varepsilon(n))\text{-secure against } \mathcal{S}^{\mathcal{P}}\right] \geq 1-\gamma$$

where $\gamma = \gamma(n)$ is a parameter, and $b = b(n)$ bounds the size of C that $\mathcal{G}$ outputs on input length n.

Note that now, when we want to use the generic transformation of Theorem IV.2 for a reduction with a probabilistic generator a la Property V.2, we need for the PCP proof to depend on the hard candidate C. To do this, we modify $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ so that in Step 1, the PCP verifier will choose a random $C \sim \mathcal{G}(1^n)$. Since this candidate C is not fixed in advance, it will be included in every PCP query made by the verifier (recall that C's size is bounded by $b(n)$). This increases the PCP length by a $2^{b(n)}$ multiplicative factor. We formalize this as a generalization of Theorem V.3.

Theorem V.3. *Let $\mathcal{R} = (\mathcal{P}, \mathcal{V}, \mathcal{S}, (\mathcal{C}, \mathcal{T}), \kappa(\cdot), \varepsilon(\cdot))$ be a reduction from a cryptographic primitive specified by $(\mathcal{C}, \mathcal{T})$ to an argument system for a language $\mathcal{L}$ as in Definition III.2. Suppose furthermore that $\mathcal{R}$ satisfies Property V.2 and has a probabilistic generator $\mathcal{G}$ for hard candidates that has output length bounded by $b(n)$ and with parameter $\gamma(\cdot)$ such that for all n, $\gamma(n) \leq 1/4$.*

Let $c = c(n)$ and $s = s(n)$ be the completeness and soundness of the argument system, and take $\varepsilon = \varepsilon(n)$, $\kappa = \kappa(n)$, $b = b(n)$, and $\gamma = \gamma(n)$. Recall that $v = v(n)$ bounds the communication from $\mathcal{V}$ to $\mathcal{P}$, the value $u = u(n)$ bounds $\mathcal{P}$'s answer lengths, the value $r = r(n)$ bounds the number of rounds, and $q = q(n)$ bounds the number of $\mathcal{P}^*$-queries made by $\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*}}(C, 1^n, 1^{\lceil 1/\varepsilon \rceil})$.

Then $(\mathcal{P}_{PCP}, \mathcal{V}_{PCP})$ is a PCP for $\mathcal{L}$ with completeness $c - \alpha - \gamma$ and soundness $\max\{s, \alpha\}$. The number of queries is $r + O[\log(1/\alpha) \cdot q]$. The alphabet size is 2^u. The length of the PCP is 2^{v+b}. Furthermore, the PCP oracle can be constructed in time polynomial in that of $\mathcal{P}(C, x, w)$ and $\mathcal{G}(1^n)$. The running time of the PCP verifier is polynomial in that of $\mathcal{G}(1^n)$, of $\mathcal{V}(C, x)$ and of $\mathcal{T}^{\mathcal{S}^{\mathcal{P}^*}}(C, 1^n, 1^{\lceil 1/\varepsilon \rceil})$.

Bounded-Adversary PRFs. In Section IV-B we obtained PCPs based on cryptographic assumptions. As noted previously, however, the type of hardness we need is much more relaxed than what is usual in the cryptographic setting: we only need hardness for a *specific* algorithm $\mathcal{S}^{\mathcal{P}}$. In this setting, for algorithms that access C as a black box, we can even obtain unconditional results. For example, to an algorithm that makes only q oracle queries, a q-wise independent hash function "looks like" a truly random function. We can use this intuition to transform (black-box) constructions of arguments from collision-resistant hash families (CRHFs) or one-way functions into PCPs unconditionally or under relatively mild complexity assumptions. The price we pay beyond the (conditional) results of Section IV-B, is that the hash function description, and with it the PCP length and verifier running time, may become large.

We define bounded-adversary pseudorandom functions. These are function families that (from black-box access) look random to a bounded adversary. We will then show that bounded-adversary PRFs (i) suffice for building a candidate generator instantiating the generic construction of Theorem V.3 and constructing PCPs from black-box reductions from one-way functions, pseudorandom function families and collision resistant hash families to arguments (ii) can be constructed unconditionally or under weak worst-case complexity assumptions (with various seed lengths).

We proceed in Section V-A with definitions and constructions of bounded-adversary cryptographic primitives. In Section V-B we show how to use the bounded-adversary primitives to transform reductions into PCPs.

A. Bounded-Adversary PRFs

We begin by defining and constructing bounded-adversary pseudorandom function families (PRFs). These are collections of functions that look random to bounded adversaries. We do not bound the time needed to compute the function: it may take more time than the adversary's running time. We construct such functions unconditionally and also under mild complexity assumptions (to decrease the function's description size).

Definition V.4 (Pseudorandom Function)**.** Consider an efficiently constructible ensemble $\mathcal{F} = \{f_n :$

$\{0,1\}^{j(n)} \times \{0,1\}^{k(n)} \to \{0,1\}^{\ell(n)}\}_n$ with seed length $j(n)$, input length $k(n)$ and output length $\ell(n)$. We say that $\mathcal{F}$ is a $(s(\cdot), \varepsilon(\cdot))$-*pseudorandom function (PRF)* if for every (non uniform) size $s(n)$ adversary (an oracle circuit ensemble) $\mathcal{A}$, for all but finitely many input lengths:

$$|\Pr_{seed \in \{0,1\}^{j(n)}}[\mathcal{A}^{f_n(seed,\cdot)}(1^n) = 1] -$$
$$- \Pr_{\text{random function } r}[\mathcal{A}^r(1^n) = 1]| \le \varepsilon(n)$$

I.e. no size s adversary can distinguish a random function in the family from a truly random function (except with advantage ε).

Note here that we do not bound the complexity of *computing* the pseudorandom functions, and in particular the function might not be computable by size s circuits. This is similar to complexity-theoretic pseudorandom generators such as that of Nisan and Wigderson [NW]. We outline several constructions of bounded-adversary pseudorandom and collision resistant functions. The first is an unconditional construction of PRFs that uses a large seed. The second construction replaces the large seed with a non uniform construction with a short seed. Then we show how to shorten the seed without resorting to non uniformity by derandomizing the unconditional construction, using the pseudorandom generators of [NW], [IW].

Proposition V.5. *For any input and output lengths $k(n)$ and $\ell(n)$, there exists a $(s(n), 0)$-pseudorandom function. The seed length is $j(n) = 2s(n) \cdot \max(k(n), \ell(n))$. The function can be evaluated in (uniform) time* $\text{poly}(s(n), k(n), \ell(n))$.

The main disadvantage of this construction is the large seed length (as large as the adversary's size). We can reduce this seed length by using non uniformity:

Proposition V.6. *For any input and output lengths $k(n)$ and $\ell(n)$, there exists a $(s(n), 1/s(n))$-pseudorandom function. The seed length is $j(n) = O(\log(s(n)))$. The function can be evaluated in* non uniform *time* $\text{poly}(s(n), k(n), \ell(n))$. *(In fact, a random advice string of this length will yield a PRF of these parameters with probability at least $1-2^{-s(n)}$. Hence, this can be viewed as a construction in the Common Random String (CRS) Model.)*

Another way of reducing the seed length without resorting to non uniformity is derandomizing. We can use derandomization techniques, e.g. the work of Impagliazzo and Wigderson [IW], to reduce the seed length without hurting pseudorandomness too much. To do this we must make mild (worst-case) complexity assumptions, this approach is taken in Proposition V.7.

Proposition V.7. *Assume that for some constant $\beta > 0$, it holds that $DTIME(2^{O(n)}) \not\subseteq SIZE(2^{\beta \cdot n})$. Then for any input and output lengths $k(n) \le s(n)$ and $\ell(n) \le s(n)$, there exists a $(s(n), 1/s(n))$-pseudorandom function. The seed length is $j(n) = O(\log(s(n)))$. The function can be evaluated in (uniform) time* $\text{poly}(s(n), k(n), \ell(n))$.

Remark V.8. *In Proposition V.7 we made a relatively strong complexity assumption, i.e. we assumed that $DTIME(2^{O(n)}) \not\subseteq SIZE(2^{\beta \cdot n})$. In general, we could use more relaxed assumptions to obtain a weaker derandomization and longer seed length. For clarity, we focus only on the "high-end" assumption made above. Note that cryptographic pseudorandom functions or collision-resistant functions are in general a stronger assumption than the assumptions needed for a derandomization a la Proposition V.7. If there exist such functions with seed, input and output length $O(\kappa(n))$, then $DTIME\left(2^{O(\kappa)}\right) \not\subseteq SIZE(poly(n))$. Using [IW] this implies the existence of* $(\text{poly}(n), 1/\text{poly}(n))$-*pseudorandom functions with similar input, output and seed lengths.*

B. Instantiations Using Bounded Adversary Primitives

Bounded-adversary PRFs can be used to transform reductions from one-way functions, PRFs or CRHFs to argument systems into PCPs. This is done by showing that for any such reduction, the bounded-adversary PRF can be used to obtain a generator $\mathcal{G}$ satisfying Requirement V.2. The fixed and bounded adversary for this PRF is the reduction run with the honest prover: $\mathcal{S}^{\mathcal{P}(\cdot,x,w)}(\cdot, x)$. Recall that this procedure makes at most $t(n)$ oracle queries to the PRF.

For reductions from one-way functions (say functions from $\{0,1\}^\kappa$ to $\{0,1\}^\kappa$), the generator simply outputs a bounded-adversary PRF chosen at random from the family with input and output length κ. Since $\mathcal{S}^{\mathcal{P}(\cdot,x,w)}(\cdot, x)$ cannot distinguish (from its bounded oracle access) this function from a random one, w.h.p. it cannot invert the function on a random input.

For reductions from CRHFs or PRFs (say with input length 2κ and output length κ) to an argument system, we use a bounded-adversary PRF family $\{f_s : \{0,1\}^\kappa \times \{0,1\}^{2\kappa} \to \{0,1\}^\kappa\}$ with seed length $j(n)$ bits. The generator $\mathcal{G}$ outputs a random member of the bounded-PRF family by choosing a random $j(n)$-bit seed. We interpret this as a PRF or CRHF by parsing the first input argument as the index to a function in the PRF or CRHF, and the second argument as the actual input.

Now since $\mathcal{S}^{\mathcal{P}(\cdot,x,w)}(\cdot,x)$ cannot distinguish f_s from a truly random function (from its bounded oracle access), it should not be able to find collisions on a random f_s. This gives us a generator for reductions from CRHFs (or PRFs).

Claim V.9. *Let $\mathcal{R} = (\mathcal{P}, \mathcal{V}, \mathcal{S}, (\mathcal{C}, \mathcal{T}), \kappa(\cdot), \varepsilon(\cdot))$ be a black-box reduction from a one-way function $f_n : \{0,1\}^\kappa \to \{0,1\}^\kappa$ or from a CRHF (or PRF) $f_n : \{0,1\}^\kappa \times \{0,1\}^{2\kappa} \to \{0,1\}^\kappa$ to a computationally sound argument system and $\gamma(\cdot)$ a parameter. Let $\mathcal{F}$ be a $(\lambda \cdot t(n) \cdot \log(1/\gamma) \cdot 1/\varepsilon^2, \delta)$-PRF as in Definition V.4, with seed length $j(n)$, input length $3\kappa(n)$ and output length $\kappa(n)$, where $\delta = 1/2 \cdot (\gamma \cdot \varepsilon - t(n)^2/2^\kappa)$ and $\lambda > 0$ is a fixed constant. Then $\mathcal{R}$ satisfies Property V.2, with the given $\gamma(n)$ and where the number of coins used by the generator $\mathcal{G}$ is $j(n)$.*

Instantiating the Generic Transformation. Recall from Sections IV-A, IV-B the generic transformation of Theorem V.3 and also the parameters it gives on known reductions. We instantiate this transformation, transforming reductions from random-oracles of collision-resistant hash families into PCPs, using bounded-adversary PRFs. Note that in this setting it even makes sense to consider reductions with logarithmic security parameter (logarithmic in the running time of the bounded adversary).

Unconditional PRF. Any reduction from one-way functions, PRFs or CRHFs to arguments yields (unconditionally) a PCP using the bounded-adversary PRFs of Proposition V.5 together with Claim V.9. The completeness, soundness, query complexity and alphabet size are as in the theorem statement or Theorem V.3. The main "price" of this instantiation is the length of the PCP that is obtained. The number of bits needed to choose a function in the family is $O(t \cdot \kappa)$, where t is the size of $\mathcal{S}^{\mathcal{P}}$ (e.g. $\mathrm{poly}(n)$ for arguments with efficient provers). The length of the PCP is thus exponential: $2^{v+O(t\cdot\kappa)}$. While this length is large, constructing even such exponential length PCPs from scratch (e.g. the Hadamard PCP of [ALM+]) is highly non-trivial. Another disadvantage is that the verifier's running time becomes quite large (polynomial in t). The PCP length can be improved either using non uniformity or derandomization, as we now describe.

Nonuniform Unconditional PRF. Continuing the discussion above, if we use the nonuniform bounded-adversary PRF of Proposition V.6, the number of bits needed to choose a function in the family becomes only $O(\log t)$. The length of the PCP shrinks to $2^v \cdot \mathrm{poly}(t)$. The verifier's running time, however, remains polynomial in t, and moreover the prover and verifier are now non uniform. An alternative approach that avoids the non uniformity (at the cost of making complexity assumptions) is derandomization.

Derandomized Conditional PRF. The final approach we suggest for transforming reductions into PCPs is shortening the seed length of PRFs using derandomization under worst-case complexity assumptions. If we assume that for some $\beta > 0$, it holds that $DTIME(2^{O(n)}) \not\subseteq SIZE(2^{\beta\cdot n})$, then we can use the PRF of Proposition V.7. The number of bits needed to choose a function in the family becomes only $O(\log t + \log(1/\varepsilon))$.[8] As before, the length of the PCP shrinks to $2^v \cdot \mathrm{poly}(t, 1\varepsilon)$. The verifier's running time, while uniform, still remains polynomial in t.

Remark V.10. *If we are willing to make stronger assumptions, we can assume here the existence of one-way permutations $f : \{0,1\}^\kappa \to \{0,1\}^\kappa$ that are hard to invert for circuits of size $2^{\Omega(\kappa)}$. This would give (using the works of [GL], [BM], [GGM]), a $(s, 1/s)$-PRFs with seeds of length $O(\log(s))$that can be computed in (uniform) time $\mathrm{poly}(\log s(n))$ and give both short PCP length and efficient verifier running time.*

VI. Acknowledgements

We thank Oded Goldreich for illuminating conversations and encouragement, Luca Trevisan for an old discussion which led to the bounded-adversary pseudorandom functions we use in Section V, and the anonymous CCC reviewers for their helpful comments.

References

[ALM+] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998.

[AS] S. Arora and S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[Bar] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *FOCS*, pages 106–115, 2001.

[BG] B. Barak and O. Goldreich. Universal Arguments and their Applications. In *IEEE Conference on Computational Complexity*, pages 194–203, 2002.

[8] Note that we could make milder assumptions about the hardness of $DTIME(2^{O(n)})$ and obtain weaker derandomizations (i.e. longer seed).

[BGS] M. Bellare, O. Goldreich, and M. Sudan. Free Bits, PCPs, and Nonapproximability-Towards Tight Results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[BGKW] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions. In *STOC*, pages 113–131, 1988.

[BGW] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In *STOC*, pages 1–10, 1988.

[BS] E. Ben-Sasson and M. Sudan. Short PCPs with Polylog Query Complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[BM] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[BCC] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[CGH] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CCD] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols (Extended Abstract). In *STOC*, pages 11–19, 1988.

[Dam] I. Damgård. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In *CRYPTO*, pages 445–456, 1991.

[Din] I. Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3):12, 2007.

[FGL$^+$] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive Proofs and the Hardness of Approximating Cliques. *J. ACM*, 43(2):268–292, 1996.

[FRS] L. Fortnow, J. Rompel, and M. Sipser. On the Power of Multi-Prover Interactive Protocols. *Theor. Comput. Sci.*, 134(2):545–557, 1994.

[Gol1] O. Goldreich. *The Foundations of Cryptography - Volume 1*. Cambridge University Press, 2001.

[Gol2] O. Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge University Press, 2004.

[GGM] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GL] O. Goldreich and L. 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, 1989.

[GMW1] O. Goldreich, S. Micali, and A. Wigderson. Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design (Extended Abstract). In *FOCS*, pages 174–187, 1986.

[GMW2] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *STOC*, pages 218–229, 1987.

[GVW] O. Goldreich, S. P. Vadhan, and A. Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof-Systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[IW] R. Impagliazzo and A. Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *STOC*, pages 220–229, 1997.

[IKO] Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Efficient Arguments without Short PCPs. In *IEEE Conference on Computational Complexity*, pages 278–291, 2007.

[Kil] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments (Extended Abstract). In *STOC*, pages 723–732, 1992.

[Mic] S. Micali. CS Proofs (Extended Abstracts). In *FOCS*, pages 436–453, 1994.

[Nao] M. Naor. On Cryptographic Assumptions and Challenges. In *CRYPTO*, pages 96–109, 2003.

[NW] N. Nisan and A. Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[RTV] O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of Reducibility between Cryptographic Primitives. In *TCC*, pages 1–20, 2004.

[Yao] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.