Probabilistically Checkable Proofs with Zero Knowledge

Joe Kilian [*] Erez Petrank [†] Gábor Tardos[‡]

Abstract

We construct PCPs with strong zero-knowledge properties. First, we construct polynomially bounded (in size) PCP's for NP which can be checked using poly-logarithmic queries, with polynomially low error, yet are statistical zero-knowledge against an adversary that makes U arbitrary queries, where U can be set to any polynomial. Second, we construct PCPs for NEXPTIME that can be checked using polynomially many queries, yet are statistically zero-knowledge against any polynomially bounded adversary. These PCPs are exponential in size and have exponentially low error. Previously, it was only known how to construct zero-knowledge PCPs with a constant error probability.

In the course of constructing these PCP's we abstract a tool we call *locking systems.* We provide the definition and also a locking system with very efficient parameters. This mechanism may be useful in other settings as well.

1 Introduction

1.1 Robust PCPs

There are quite natural parallels and explicit transformations between multi-prover interactive proof systems (MIPs) and probabilistically checkable proofs (PCPs). However, zero-knowledge does not seem to be conserved under these transformations, and indeed the very definition of zero-knowledge for PCPs requires some care. In the MIP framework, a prover can for example simply refuse to answer a question that would not have been asked by the honest verifier. However, PCPs are formally viewed as sequences of bits; there is no entity in place to judge a question's legitimacy. Consequently, the theorem that everything provable in MIP is provable in zero-knowledge [2, 7] does not translate over automatically.

Dwork et al. [3] introduce the notion of zero-knowledge or "robust" PCP's. A robust PCP (Ω, V) is a distribution Ω on PCPs along with a probabilistic polynomial-time verifier V. The prover samples a PCP from the given distribution, writes this sample down, and the verifier checks the sample as it would any PCP. (Ω, V) has two new parameters, L and U, with $L \leq U$: V can check Ω by querying L bits, but one can simulate in probabilistic polynomial time the view of any adversary $\hat{V}$ who adaptively reads up to U arbitrary bits. As with other models, the quality of the simulation determines the type of zero-knowledge obtained.

For $L \geq 3$ and U essentially arbitrary, Dwork at al. exhibit a robust PCP (Ω, V) for NP. However, the error obtained by these protocols (the probability that V will accept a false statement) is as big as $1 - \Theta(1/U^3)$. Thus, the construction of [3] left open the following question:

[*] NEC Research Institute. E-mail: joe@research.nj.nec.com

[†] DIMACS Center, P.O.Box 1179, Piscataway, NJ 08855. Email: erez@dimacs.rutgers.edu. Part of this work was done while the author was at Toronto University and while the author was visiting the NEC Research Institute.

[‡] Mathematical Institute of the Hungarian Academy of Sciences, Pf. 127, Budapest, H-1364 Hungary and Institute for Advanced Study, Princeton, NJ 08540. Supported by NSF grants CCR-95-03254 and DMS-9304580, a grant from Fuji Bank and the grant OTKA-F014919. E-mail: tardos@cs.elte.hu.

Question: Does there exist a robust PCP with error less than $\frac{1}{3}$?

Indeed, the best previous PCPs had error $1 - c$ for some small constant c. Note that the standard error-reduction techniques do not work here, since they require the verifier to query more bits than before, yet leave U unchanged. As a result, L becomes greater than U.

Robust PCPs and the hardness of approximation

Our motivation for this investigation is primarily philosophical. However, it is worth noting that constructions for robust PCPs, or variants thereof have been used to prove results on the hardness of approximation. Kilian and Naor [8] directly use robust PCPs to establish the strongest known hardness of approximation results for certain problems in statistical inference. Feige and Kilian [4] use PCPs inspired by robust PCPs, though with slightly weaker properties, to show the strongest known results for approximating the chromatic number.

1.2 The results in this paper

We exhibit new constructions for robust PCPs. The properties of our first construction is embodied in Theorem 1, below. In a nutshell, Theorem 1 says that there exists robust PCPs in which the number of queries U needed to break the statistical zero-knowledge property is nearly exponential in the number of queries L needed to check the proof. For a reasonable setting of the parameters in Theorem 1 and for proofs of NP, this would mean having proofs of size polynomial in the length of the input n, having $L = \text{poly}\log n$ (i.e., L is poly logarithmic in the length of the input) and $U = n^c$. Using k repetitions of the basic protocol, we can achieve 2^{-k} error with $L = k\text{poly}\log n$ and $U = n^c$.

Theorem 1 *Let PCP (Π, V) for a language L have size at most n and require the verifier to use at most $\log n$ coin tosses, checks at most a constant number of bits in Π and achieves a constant soundness error probability (smaller than 1). Let $U = n^{O(1)}$. Then there exists a robust PCP (Ω, V') for L with the following properties:*

- **Size of proof:** *For any $\Pi \in \Omega$, $|\Pi| = O(n^{O(1)})$.*
- **Cost of verification:** *The honest verifier V' reads $L = O(\log^{O(1)} n)$ bits of the proof, and tosses $O(\log^{O(1)} n)$ coins.*
- **Completeness:** *If the original proof Π has perfect completeness with respect to V, then so does any $\Pi' \in \Omega$ with respect to V'.*
- **Soundness:** *For any Π' (not necessarily in Ω), the probability that the verifier accepts an $x \notin L$ is at most $1/n^{O(1)}$. (Increasing the constant in the power of n is linearly related to increasing the constant in the cost of verification.*
- **Zero-Knowledge:** *Ω achieves statistical zero-knowledge against any verifier V' that is allowed to read at most U bits of Π' (uniformly selected in Ω). More precisely, there is a probabilistic polynomial time simulator S (with an oracle for V') that simulates V''s view of Π'. This simulation differs (in the ℓ_1-norm) from V''s actual view by less than $1/n^c$ for any c, as n grows sufficiently large.*

Theorem 1 can be generalized to handle any number of bits read by the original PCP, and any number of random coin tosses by the original verifier. We chose to keep the complexity of the statement reasonable and still to cover the PCP constructed by the PCP theorem [1].

Our second construction allows the PCPs to be exponentially large, but achieves superpolynomial robustness (with an exponentially small error). Without going into all of the parameters, we obtain the following result.

Theorem 2 *For any language $L \in \mathsf{NEXPTIME}$, there exists a robust PCP (Ω, V) for L with the following properties.*

- **Cost of verification:** *On input x, V checks Ω using polynomially many (in $|x|$) queries, computations and coin tosses.*
- **Completeness:** *(Ω, V) has perfect completeness.*

- **Soundness:** *For any Π' (not necessarily in Ω), the probability that the verifier accepts an $x \notin L$ is at most $2^{-|x|}$.*

- **Zero-Knowledge:** *Ω achieves statistical zero-knowledge against any polynomially bounded verifier.*

1.3 Locking systems

In the course of constructing the PCP for NP, we use a locking system with very efficient parameters. A locking system is a pair of efficent probabilistic machines: The locker and the checker. The locker is given a string S to "hide" and produces a string L (the lock) and a key K to the lock. The lock has the following properties: It is "hard" to find the string S hidden in the lock. Given the key K, it is "easy" to reconstruct the secret S and check that this is indeed the hidden string in the lock. Last, only one string can be hidden in a lock. A legitimate verifier will reject, with high probability, any other string no matter what key he might possess. Hardness and easiness in this settings refer to the number of bits in L that have to be read. For the exact definition and construction see Section 3.

1.4 Guide to the rest of the paper.

In Section 2 we give the basic building blocks of our constructions. In Section 3 we provide the definition of locking systems and construct a locking system with efficient parameters. In Section 4 we use the efficient locking system to prove Theorems 1 and 2.

2 PCPs robust against random queries

Suppose one has a PCP (P_0, V_0) for a language L such that

- $|P_0| = n$,

- V_0 flips $O(\log n)$ coins and queries $O(1)$ bits of P_0,

- (P_0, V_0) has perfect completeness, and

- V_0 has error bounded from 1 by a constant.

(Note that n here refers to the size of the proof and may be exponential in the size of the input.) Then the techniques of [3] allow us to transform (P_0, V_0) into a new PCP (Ω, V) with the following properties, where $k > 0$ is an integral parameter:

- Any proof assigned nonzero probability by Ω has size $O(k^3 n^{O(1)})$,

- V flips $O(\log n)$ coins and queries 3 bits of the proof.

- The proof has perfect completeness and soundness error at most $1 - 1/O(k^3)$.

- Ω is robust against k arbitrary queries. That is, the view so obtained can be perfectly simulated in probabilistic polynomial time.

Note that the soundness of the protocol degrades super-linearly with respect to its robustness (k). Thus, if one tries to reduce the error to less than $\frac{1}{2}$ by making multiple queries to the PCP then the number of queries so made would exceed the proof's robustness.

As a useful special case, we consider the verifier that simply runs the original verifier m times, with independent coin toss sequences, and makes all of the $3m$ queries specified by these runs. We achieve statistical zero-knowledge against such verifiers as follows. Given a DFKNS-type randomized PCP P, consider the randomized PCP $P' = P^{\ell}$ generated by concatenating ℓ PCPs, each independently chosen according to P. The verifier V' first chooses i uniformly from $\{1, \ldots, \ell\}$ and then runs the original verifier V on the ith copy of the proof.

Suppose that P is robust against $3k$ queries. Then as long as a verifier directs at most k 3-bit queries to any subproof of P', the view obtained will be easily simulatable. If one makes m independent random runs from V' then the probability that any one subproof is selected by V' more than k times is bounded above by

$$\ell \binom{m}{k+1} / \ell^{k+1} < \frac{m^{k+1}}{l^k}.$$

If we set $\ell = m^2$, the above expression is at most $1/m^{k-1}$. Thus, when k grows as a super-constant function of n, (P', V') is *statistical* (not perfect) zero-knowledge against a verifier that makes m independent runs of V'. The crux of our main constructions is to code P' in such a way that even an arbitrary verifier can only ask essentially random questions.

Lemma 2.1 *Let k and m be arbitrary parameters and let (P_0, V_0) be as above. We can transform (P_0, V_0) into a new PCP (Ω, V) such that*

1. *Any proof assigned nonzero probability by Ω has size $O((kmn)^{O(1)})$,*
2. *V looks at 3 bits of the proof.*
3. *V tosses at most $O(\log n + \log k + \log m)$ coins*
4. *The proof has perfect completeness and the new verifier has error at most $1 - 1/O(k^3)$.*
5. *If some verifier V' makes $3m$ random queries chosen by m independent runs of V, then with probability $1 - 1/m^k$ V''s view will be simulatable in polynomial time. More precisely, whether V''s view is simulatable is a function of which location V' queries. V' will query a "hard" set, in which the simulator cannot perfectly simulate the results of the query, with probability at most m^{-k}.*

By making m polynomial in n and k superconstant, we obtain polynomial size PCPs that are statistically zero-knowledge against a large (polynomial) number of random queries. My making m exponential in n we obtain exponential-sized PCP's that are statistically zero-knowledge against any polynomial number of queries.

3 Locking systems

In this section we define locking systems and construct a lock which we later use in our PCP system for NP. Loosely speaking, a lock is a string which holds a secret. On one hand, we would like the non-legitimate user to find it "hard" to discover the secret locked in the string. A second demand is that given the key to the lock, a verifier can "easily" reconstruct the secret and verify that this is the secret hidden in the lock. The terms "easy" and "hard" relate to the number of bits that have to be read from the lock in order to perform the task. Last, we require that the lock will be a commitment on one secret. Namely, there is only one string that the verifier accept with high probability (when given the right key to the lock). For any other string and for any possible key, the verifier will reject with high probability. For simplicity we require that the key determines the secret in deterministic polynomial time without the help of a lock. Also, for simplicity we do not allow several keys determining the same secret to fit a lock.

In what follows, we use n as a security parameter given to the locking system. In the application, n will be the size of the PCP proof, and this role of n matches the definitions in the rest of the paper. The formal definition follows:

Definition 3.1 (A locking system): *Let $\mathcal{S} = \{0,1\}^s$ be the set of secrets, $\mathcal{L} = \{0,1\}^l$ be the set of locks and $\mathcal{K} = \{0,1\}^k$ the set of keys. Here the parameters s, l, k (and thus the sets $\mathcal{S}$, $\mathcal{L}$, $\mathcal{K}$) depend on the security parameter n. A* locking system *is a pair of probabilistic polynomial time procedures, the* locker *and the* checker*. The locker takes a secret $S \in \mathcal{S}$ as input and produces a lock $L \in \mathcal{L}$ and a key $K \in \mathcal{K}$. The checker takes a key $K \in \mathcal{K}$, reads some bits of a lock $L \in \mathcal{L}$ and either accepts, or rejects. We say that the locker and the checker form a locking system with error $\epsilon(n)$, robustness $U(n)$, and checkability $C(n)$ if the following conditions hold.*

1. perfect completeness: *There is deterministic polynomial time procedure to find the secret S from the key K produced by the locker on input S. If L and K are produced by the locker on input S, then the checker on input L, K reads at most $C(n)$ bits from L and accepts with probability 1.*

2. Soundness (or a commitment property): *For every lock $L \in \mathcal{L}$ there is at most a single fitting key K. For all other keys $K' \neq K$ the checker rejects with probability at least $1 - \epsilon(n)$.*

3. Secrecy (indistinguishability): *For any secret $S \in \mathcal{S}$, let L_S be the distribution of the locks output by the locker on S, then for any $S, S' \in \mathcal{S}$ the two distributions L_S and $L_{S'}$ are statistically indistinguishable for a machine that can read only $U(n)$ bits from a sample. Namely, for all probabilistic (computationally unbounded) Turing machines A, for all constants c and sufficiently large n, and for all pairs $S, S' \in \mathcal{S}$,*

$$|\mathrm{Prob}_{L \in L_S}[A(L) = 1] - \mathrm{Prob}_{L \in L_{S'}}[A(L) = 1]| < \frac{1}{n^c}$$

4. Zero knowledge (simulatability): *There exists a probabilistic polynomial time procedure that produces answers to at most $U(n)$ bit-queries to a (non-existing) lock and then as it receives any secret S it produces a valid lock, key pair for this secret with the lock also fitting the previous answers. Far all probabilistic (computationally unbounded) Turing machines A asking the bit queries and for all secrets S this procedure yields a distribution on locks which statistically indistinguishable from the distribution of the locks produced by the locker on input S.*

Note that simulability implies indistinguishability, we listed the latter property separatly to emphasize the similarity to bit commitments.

Theorem 3 *Let $t = O(\log n)$ and let q be an unbounded (non-constant) function such that $2^t > 5q$. There exists a locking system in which the locker locks secrets of length qt, outputing locks of length $q2^{2t+1}$ and keys of length $2qt$. The locking system has error $\epsilon = 15/16$ robustness $U(n) = \sqrt{2^t/n}$ and checkability 2.*

Remark 3.2 *Note that if we let the verifier access the lock ℓ times, we get a system with the same robustness, checkability 2ℓ and error $(15/16)^\ell$.*

In the rest of the section we prove Theorem 3.

3.1 A basic construction

Inuitively, the main idea of our construction is to use directions in the plane. Fix a direction of a finite plane, and choose a binary function which is a constant along each line in this direction. The direction will be the secret and the key, and the lock will be the truth-table of the function. In the next paragraph, we implement this idea in details, but before doing that note that this has the flavor of a locking system. In order to discover the hidden direction without having the key, one must read "lots" of points from the function, whereas given the direction, one may "easily" check that this direction is special for this function. Let us proceed and implement this idea while stressing that as is, this idea is not enough to achieve a locking system.

Consider a polynomial size finite field F which will also function as the set of all possible secrets $\mathcal{S}$. To hide an element $S \in F$ choose a random function $r : F \to \{0,1\}$ and output the truth table of the function $f : F^2 \to \{0,1\}$ defined by $f(a,b) = r(aS - b)$. Given S, a spotcheck is verifying $f(a,b) = f(a+t, b+St)$ for a random triplet $(a,b,t) \in F^3$. Completeness-wise it is easy to check that the spotchecks must hold for any valid lock for S. In terms of soundness, a deceiving lock has a good chance for the spotchecks on f to hold for two different values S and S' only if f is almost a constant (we prove this later in the special case we need). So in order to verify that a locker L hides the secret S, the checker has to make a constant number of spotchecks, and to reject if one of them fails or if the values he gets do not seem to be uniformly distributed. After doing this, the verifier can be convinced that the lock is committed indeed to S with high (yet constant) error probability $1 - \epsilon$.

In terms of indistinguishibility, this system is not secure even against a checker that reads two points out of the lock. With probability $1/|F|$ the checker looks at the right direction and gets positive indication to that. However, this construction seems to be a first step in the right direction since with "quite good" probability the checker will have to read "many" points before encountering two points of the predetermined direction S. And before encountering two points on direction S, the checker only sees uniformly chosen random bits (the function r). We shall improve the security parameters later.

3.2 Drawbacks of the basic constructions

Let us list the drawbacks of this primitive lock and then proceed with solving them.

1. **Indisinguishibility not achieved:** The role of the key and the secret are both played by the value S. Thus, if one suspects the value of the secret it can be easily checked. We want that even if one knows that the secret is one of two possible values, limited access to the valid lock should not help him decide which value is correct. Furthemore, even if one does not know the value of S he might still guess and check. For a key of logarithmic length (which is a requirement if the truth table is to be polynomial in size), the probability of success is not negligible. To summarize, indistinguishibility is not achieved.

2. **Completeness is not perfect:** Finally, one prefers perfect completeness, i.e. no chance of rejection for a valid lock. Here an unlucky checker can reject after seeing only zeros of a valid lock (an unlucky legitimate locker can even produce a constant f that is always rejected).

Before presenting our locking system we give an intuitive account of how we solve both of the above problems.

We solve the first problem by using many copies of the basic construction to hide digits of a Reed-Solomon codeword. We make sure that breaking a few of them does not compromise the secret (in the strong sense of indistinguishibility and simulatability) and if one reads only $U(n)$ bits of the secret, then one can break many of them with negligible probability only.

To solve the second problem we make a stronger demand on a valid lock: the function f must evaluate to zero on exactly half of the points in the plane. This allows not to reject in case the checker sees only zeros or only ones in the original spotchecks. Specificallly, we choose the field F of characteristic 2 and insist that the function $r : F \to \{0,1\}$ used in locking a secret x satisfies $r(b) = \overline{r(b+1)}$ (defining $\overline{x} = 1-x$) for every value $b \in F$. Note that $(b+1)+1 = b$. We choose a random r which satisfies this restriction. This implies that a valid lock L satisfies $L(a,b) = \overline{L(a,b+1)}$ for every $a, b \in F$ and it is enough to specify only one of these values in the lock, it determines the other. Thus, by specifying only half of the lock, we make sure that the even invalid locks satisfy the balancing property, ensuring the soundness property of the lock. We proceed with the formal construction and proof.

3.3 Constructing a locking system with good parameters

Let us describe our system in detail. Let $q = q(n) = \omega(1)$ be any (polynomially bounded, polynomial time computable) parameter. We will use $4q$ basic systems in the construction. Let F be a finite field of size $|F| = 2^t$ with $t = \Theta(\log n)$ and $|F| \geq 5q$. Let C and D be two disjoint subsets of F with $|C| = q$ and $|D| = 4q$ (think of C as being the first q elements in some enumeration over the elements of F, and D being the next $4q$ elements). Our set of keys $\mathcal{K}$ is the set of polynomials over F with degree smaller than $2q$. Thus the binary length of a key is $k = 2qt$. The secret determined by a key p is $p|C$ i.e. the values of the polynomial p on the q elements in C. Thus the binary length of the secret is $s = qt$. Note that the set $\mathcal{S}$ of all secrets is the set of all functions from C to F.

The locker: To lock a secret $S : C \to F$ the locker starts by choosing a uniform random polynomial p of degree less than $2q$ with $p|C = S$. Then, the locker uses the basic lock to lock the value $p(d)$ for each $d \in D$. Specifically, the locker chooses a uniform random function $r_d : F \to \{0,1\}$ satisfying $r_d(b) = \overline{r_d(b+1)}$ for every $d \in F$, and sets a basic lock $f_d(a,b) = r_d(p(d)a - b)$. Recall that this construction ensures that $f_d(a,b) = \overline{f_d(a,b+1)}$ for every a, b and d. Thus, it is enough to store in the basic lock only the value of one of them, and if needed we compute the other. Finally, we define the lock to be the set of basic locks f_d for all $d \in D$.

The checker: Given a key $p \in \mathcal{K}$, i.e., a polynomial over F with degree smaller than $2q$, and a lock $L \in \mathcal{L}$, the checker perfoms the following test to check that the key fits the lock. The checker picks a value $d \in D$ uniformly at random, and performs a spotcheck on the basic lock f_d: He picks a triplet $(a,b,t) \in F^3$ uniformly at random, and checks that $f_d(a,b) = f_d(a+t, b+t \cdot p(d))$.

3.4 Analysis of the locking system

3.4.1 Parameters:

Recall that $t = \Theta(\log n)$ and that q is an unbounded function. We think of q as a slowly growing function below $\log n$. A key is of length $k = 2qt$ which is slightly superlogarithmic, but $k = O(\log^2 n)$. Note that k has to be superlogarithmic in any locking system or the key can be guessed with polynomial success probability. A secret is of length qt, i.e., $s = O(\log^2 n)$. A lock is of size $4q \cdot |F|^2/2 = q2^{2t+1}$.

We remark that we chose to implement $k/s = 2$, but any constant ratio above 1 will work here. Clearly if the key is to determine the sectret the key has to be at least as long as the secret.

The number of bits of the lock the checker reads is two. This is best possible as checkability 1 implies $k \geq U$ for any locking system (where U is the robustness of the lock), meaning too long keys.

3.4.2 Completeness:

To retrieve the secret from the key one evaluates p on all points in C. Clearly, if the locker follows his protocol, and the checker gets the output lock and key, then the checker will always accept.

3.4.3 Soundness:

We show that the soundness error is at most $\epsilon \leq 15/16$. Consider the basic scheme first. We make the following claim.

Claim 3.3 *Let $f : F^2 \rightarrow \{0,1\}$ be a function which has exactly the same number of zeros and ones in its outpout. For such a function, there is at most one value $x \in F$ such that a spotcheck of f in direction x holds with probability greater than $3/4$.*

Proof: Let $x \neq y$ be two values in F. Consider the following experiment: We pick unifromly and independently at random two points $A, B \in F^2$, and consider the point C in which the line through B in direction y meets the line through A in direction x. The point C is a randomly chosen point on the line of A in direction x. Thus, comparing the value of f on A and on C is actually a uniform random spotcheck in direction x. Similarly, comparing the value of f on B and C is actually a random spotcheck of direction y. By the balance property of the function, we know that for the random points A and B it holds that with probability exactly $1/2$ that $f(A) \neq f(B)$. In this case, one of the spotchecks must fail. So with probability at least $1/2$ one of the directions fail, then one of these directions must have its spotchecks fail with probability at least $1/4$. ■

Now fix any lock $L \in \mathcal{L}$. It contains the functions $f_d : F^2 \rightarrow \{0,1\}$ for every $d \in D$ and all of these functions have as many zeros as ones. Let $g : D \rightarrow F$ be a function where $g(d)$ is the direction in which a spotcheck of f_d is most likely to pass (we break ties arbitrarily). By claim 3.3, any spotcheck on f_d with direction different than $g(d)$, fails with probability at least $1/4$. Recall that a key is a polynomial of degree smaller than $2q$. Thus, two different keys must agree on less than $2q$ points in D, and thus they disagree on more than $2q$ points. Therefore, there could be only one polynomial (i.e., only one key) which disagrees with $g(d)$ on at most q points. We claim that for all other keys, i.e., for each key $p \in \mathcal{K}$ which differs from g on more than q values of D, the checker accepts the key p on the lock L with probability less than $15/16$.

Since the checker chooses a function f_d to check uniformly in $d \in D$, then with probability over $1/4$ we have $g(d) \neq p(d)$. Then it performs a spotcheck of f_d in direction $p(d)$, and since $p(d) \neq g(d)$, we get that the checker rejects with probability at least $1/4$. Thus there is only one key for which the probability that the checker rejects is at most $1/16$.

Remark 3.4 *We have shown that the error parameter here is $\epsilon \leq 15/16$. We remark that one can decrease the value of this constant to $7/8$ but with the checker reading only a constant number of bits of the lock the error must be a constant. Performing h independent checks one can decrease the error to ϵ^h for the price of multiplying the number of bits read by the checker by h.*

3.4.4 Indistinguishibility:

Let L_S to be the distribution on locks output by the locker on a secret $S \in \mathcal{S}$. In this section we are going to show that for any two secrets $S, S' \in \mathcal{S}$, the two distributions L_S and $L_{S'}$ are statistically indistinguishable by any probabilistic machine that reads at most $U(n) \leq \sqrt{|F|/n}$ bits from a sampled lock.

Recall that the distribution L_S is output on a secret S by the locker conducting the following procedure. The locker picks uniformly at random a polynomial $p \in \mathcal{K}$ such that $p|C = S$. This polynomial is the key. Then, the locker selects $|D|$ uniform random functions $r_d : F \rightarrow \{0,1\}$ satisfying $r_d(b) = \overline{r_d(b+1)}$ for any $b \in F$, one function for each $d \in D$,. Finally, the locker sets the lock to be the set of functions $f_d(a,b) = r_d(a \cdot p(d) - b)$ for all $d \in D$.

Fix the value $m = U(n)$ (the robustness of the system). We are going to show that the distributions L_S and $L_{S'}$ are statistically indistinguishable even if we allow the machine to be more powerful then just reading bits from the lock. Suppose instead of letting the machine read m bits of the lock, we let it ask m^2 queries of the following type: The machine writes a pair (d,x), where $d \in D$ and $x \in F$, on a special query tape, and then it gets an answer whether $p(d) = x$. After making m^2 (adaptive) queries of this type and running some unbounded computation, the machine produces an output. We call this machine *the query machine*. Let us first show that the query machine is stronger. Later, we show that even a query machine cannot distinguish well between L_S and $L_{S'}$. To show this, we show that a query machine which makes m^2 queries can perfectly simulate the output distribution of a (regular) machine which reads at most m bits of the lock.

Let M be a machine reading at most m values of a lock L. We build a query machine M' that makes at most m^2 queries and produce the same distribution of outputs as M. (The distribution is on the choice of a lock $L \in L_S$ and on the choice of a random tape for M.) M' will run M and use its m^2 queries to simulate the bits read by M from the lock. M' also keeps track of M's readings so that if a bit of the lock is read more than once by M, then M' returns the same value each time. M' is going to let M get random bits for all its queries unless it reads two bits $f_d(a,b)$ and $f_d(a',b')$ for two points (a,b) and (a',b') lying on the unique direction $p(d)$. In this case, the machine M' feeds M with the same bit in both cases. Since $f(a,b) \neq f(a,b+1)$ we also must check the relation of (a,b) and $(a',b'+1)$.

This is implemented in the following manner. Whenever M asks for a new bit $f_d(a,b)$ of the lock, machine M' checks all points (a',b') on which M has queried f_d previously. If $(a,b) = (a',b')$ for some previous point, then M' gives M the same answer as before (denote by $q_d(a',b')$ previous answers given to M by M'). Similarly if $(a,b+1) = (a',b')$ for a previous point, then M' gives $q_d(a,b) = \overline{q_d(a',b')}$ to M. Otherwise, for all these points in which $a \neq a'$ M' makes queries to check if one of them satisfies $p(d) = (b-b')/(a-a')$ or $p(d) = (b-b'+1)/(a-a')$. If all queries are false, then machine M' chooses uniformly at random a bit, sets $q_d(a,b)$ to contain this bit for future use, and feeds this new bit as an answer to M. Otherwise, if there is a point (a',b') for which $p(d) = (b-b')/(a-a')$, then machine M' sets $q_d(a,b) = q_d(a',b')$ and feeds $q_d(a,b)$ to M, or if there is a point (a',b') for which $p(d) = (b-b'+1)/(a-a')$, then machine M' sets $q_d(a,b) = \overline{q_d(a',b')}$ and feeds $q_d(a,b)$ to M. (Note that in case $a = a'$ and $b \neq b'$ queries are not needed, the points are not on the same line.)

We go on with verifying the indistinguishibility by any query machine (making at most m^2 queries). Let S, S' be any two secrets in $\mathcal{S}$. We would like to show that L_S and $L_{S'}$ are indistinguishable by any query machine. For a secret $S \in \mathcal{S}$ let $\mathcal{K}_S$ be the set of keys $p \in \mathcal{K}$ corresponding to secret S, i.e. the set of all poynomials p of degree smaller than $2q$ having $p|C = S$. We have $|\mathcal{K}_S| = |F|^q$. Consider now a query machine M. We make two simplifying assumptions. First, since M is not computationally bounded, we may assume that M is deterministic (pick the best random tape for M). Thus, the run of M is determined by the sequence of answers it receives for the queries it makes. Second, we assume that M "wins" as a distinguisher (or really recovers the secret S) in case M gets to learn the value of the key p on q different elements of D.

(Clearly with one more query, he could tell between two different secrets.) Thus, in what follows, we assume that runs have at most q "yes" answers in them. After q values have been guessed correctly, the run terminates (successfully).

We would like to compute the probability that M accepts for a random key in K_S and for a random key in $K_{S'}$. Think of a run of M as a series of m^2 queries and answers made by M and answered by the oracle followed by M's accepting or rejecting. The run of M is completely determined by the answers that M gets for his queries since M does not toss coins. Let T be a run, let S be a secret, and denote by $K_{T,S}$ the set of all keys for S which are consistent with the answers given in the run. The probability that the run T occurs, given that the secret is S, is $|K_{T,S}|/|K_S|$. The distribution space is that of the locker choosing a random key in K_S. The probability that M distinguishes S from S' is:

$$\left| \sum_{\text{accepting T}} \frac{|K_{T,S}|}{|K_S|} - \frac{|K_{T,S'}|}{|K_{S'}|} \right|$$

$$\leq \sum_T \left| \frac{|K_{T,S}|}{|K_S|} - \frac{|K_{T,S'}|}{|K_{S'}|} \right|$$

We note that $|K_S| = |F|^q$ independently of S, and in what follows, we compute for each run T, a bound on the magnitude of $|K_{T,S}| - |K_{T,S'}|$.

Consider a particular run T of the machine M. By the end of the run, M's view contains the value of p on the places $d \in D'$ for a set $D' \subset D$, $|D'| \leq q$. It also contains inequalities of the form $p(d) \neq x$ for some pairs $(d, x) \in E$, where $E \subset (D - D') \times F$ and $|E| \leq m^2$. To compute $|K_{T,S}|$, we start with the set $K_{D'}$ of all keys $p \in \mathcal{K}_S$ having the required values on D', and use the inclusion-exclusion formula to remove those who are inconsistent with the inequalities discovered in the run:

$$|K_{T,S}| = \sum_{i=0}^{|E|} (-1)^i$$

$$\sum_{E' \subseteq E \, |E'|=i} |\{p \in K_{D'} | p(d) = x \text{ for every } (d, x) \in E'\}|$$

Notice that the summand is zero if E' contains two values with the same first coordinate d, and otherwise it is $|F|^{q-|D'|-i}$ if $i \leq q - |D'|$. In particular the summand for $i \leq q - |D'|$ is independent of the secret S. As the partial sums alternate in being lower and higher than $|K_{T,S}|$ this means that the difference $||K_{T,S}| - |K_{T,S'}||$ is below the absolute value of the term with $i = q - |D'|$. Namely,

$$\big||K_{T,S}| - |K_{T,S'}|\big| \leq \binom{m^2}{q - |D'|}.$$

This means that for any run in which the number of "yes" answers given by the oracle (i.e., the size of D') is i, the diffence $||K_{T,S}| - |K_{T,S'}||$ is bounded by a value that is independent of the secrets S, S' and of the run T. We are now going to sum over all runs, partitioning them by the size of D'. Note that our estimate also holds for $|D'| = q$. In this case, $0 \leq |K_{T,S}|, |K_{T,S'}| \leq 1$, and thus the difference is at most 1.

Since the run T is determined by the answers for the queries, there are at most $\binom{m^2}{i}$ runs T with i "yes" answers, that is with $|D'| = i$. Therefore,

$$\begin{aligned}
\sum_T \left| \frac{|K_{T,S}|}{|K_S|} - \frac{|K_{T,S'}|}{|K_{S'}|} \right| &= \frac{1}{|F|^q} \sum_T ||K_{T,S}| - |K_{T,S'}|| \\
&\leq \frac{1}{|F|^q} \sum_{i=0}^{q} \binom{m^2}{i} \binom{m^2}{q-i} \\
&= \frac{\binom{2m^2}{q}}{|F|^q} \\
&< \left(\frac{m^2}{|F|}\right)^q
\end{aligned}$$

Since $m \leq \sqrt{|F|/n}$, this fraction is negligible. Thus, we get that a negligible fraction is an upper bound on the distinguishibility of any two secrets S, S' by any query machine M that makes at most m^2 queries and we are done.

3.5 Simulatability

Let us further prove that for our specific locking system indistinguishibility implies simulatability. We look at the user invoking the simulator as a query machine as in the previous paragraph. Note that this is OK since the query machine can perfectly simulate the output of a regular user. The simulator works as follows. It begins by setting the secret to be the all zeros secret. Then, it uses the checker to produce a lock for this specific secret, and lets the user make U^2 queries to the key output by the locker. Later, the user comes up with a secret S and the simulator replies whith a lock and key for that secret such that the lock matches the user's view so far.

After querying the lock U^2 times, the user has some information on the key which includes some values that the polynomial must evaluate to ("yes" answers) and some values that the polynomial does not evaluate to ("no" answers). The simulator now chooses a random polynomial that matches both the information given so far on the key and the values of the polynomial as determined by the given secret S. The simulator outputs this polynomial as the key for the secret S. and produces the yet unseen part of the lock randomly according to this key.

Let us first show that such a polynomial can be chosen in probabilistic polynomial time with a very high probability of success. If the user has seen q "yes" answers, thus he knows the value of the polynomial at q different locations, then the simulation fails. This happens with negligible probability (on a random key for the all zero secret) as shown in the previous subsection. So suppose the user sees less than q values of the polynomial. To choose a proper polynomial, the simulator first chooses a random polynomial which matches the values of the polynomial known to him. If this polynomial contradicts the inequalities he knows, he tries again and again until he succeeds. Let us argue that there is a probability $1 - 1/n$ to succeed in each of these tries, thus repeating this process more than a constant times fails with negligible probability. Since the value of the polynomial is predetermined in less than $2q$ places, choosing a random polynomial that matches these less than $2q$ values leaves the value at any specific other point random. The user has seen at most U^2 inequalities ("no" answers). Each one of these inequalities is falsified by a random polynomial with probability at most $1/|F|$. Thus the probability that a random polynomial hits a forbidden value is at most $U^2/|F| \leq 1/n$.

Next, we claim that the distribution output by the simulator is indistinguishable from a "real" distribution the locker produces on the secret S. Suppose a user A could tell the difference. We think of A as a querry machine which can also look at the entire lock and key after his U^2 queries. Surely, this last stage does not help him, as in both cases the same thing happens: he receives a uniform random lock-key pair describing S and fitting his view so far. But if he distinguished the two cases before this last stage, then he actually distinguished the all zero secret from S. We know that he can do this with negligible probability only.

This completes the analysis of the locking system and the proof of Theorem 3. ■

4 Proofs of Theorems 1 and 2

In this section, we describe a technique for using locks to convert PCPs robust against random queries into PCPs robust against directed queries. This technique is then applied to the PCP of Section 2 to prove Theorems 1 and 2.

4.1 The new PCP

The structure of the new zero knowledge PCP that we build consists of three tables, PCP, PERM and MIX. We first describe how they are generated and then show how to verify them. The prover executes the following steps.

1. **Locking the PCP:** The prover generates a PCP according to Lemma 2.1. This PCP consists of an array of bits, $b_1, \ldots, b_m$. For each bit b_i the prover generates a lock/key pair (L_i, K_i) and sets $\mathsf{PCP}(i) = L_i$.

2. **Locking a random permutation of the keys:** Let R denote the space of random coin tosses used by the original verifier. The prover generates a random permutation π on R. For each $r \in R$, the prover generates a lock/key pair (L_r, K_r) for te value $\pi(r)$ and sets $\mathsf{PERM}(r) = L_r$.

3. **Allowing random access to the PCP:** The prover generates MIX by

$$\mathsf{MIX}(\pi(r)) = (r, K_a, K_b, K_c, K_r),$$

where, the original verifier would look at bits a, b and c given random string r.

We leave unspecified the parameters of the locks, which will vary depending on which theorem we wish to prove. To verify the new PCP, the new verifier V executes the following steps.

1. V chooses $r' \in R$ at random, and queries $MIX(r') = (r, K_a, K_b, K_c, K_r)$.

2. V uses K_r to unlock $\mathsf{PERM}(r)$ and checks that the resulting value is indeed r'.

3. V computes the locations (a, b, c) of the 3 bits the original verifier V_0 would have checked given r, and uses K_a, K_b and K_c to unlock $\mathsf{PCP}(a)$, $\mathsf{PCP}(b)$ and $\mathsf{PCP}(c)$, obtaining b_a, b_b and b_c.

4. V checks that V_0 would have accepted (r, b_a, b_b, b_c).

Finally, V rejects if any unlocking operation or any check fails, and accepts otherwise.

4.2 Analysis of the new PCP

We now bound the relevant parameters of the new PCP in terms of the parameters of the original PCP and the parameters used in the locks.

Perfect Completeness: It is easy to verify that if the original PCP had perfect completeness then the new PCP has perfect completeness.

Complexity: Suppose that the PCP used by Lemma 2.1 was of size n and the verifier uses $\log n$ random bits. Then the PCP constructed by Lemma 2.1 will be of size $O((kmn)^{O(1)})$, where m and k are parameters to be set later. Thus PCP will have $O((kmn)^{O(1)})$ locks. Similarly, MIX will have $O((kmn)^{O(1)})$ entries, each consisting of string of length $O(\log n + \log k + \log m)$ (to represent r) and 4 keys, and PERM will have $O((kmn)^{O(1)})$ locks. Each key is of length $O(qt)$ and each lock is of length $O(2^{2t}q)$. Hence, the size of the proof will be $O(2^{2t}q(kmn)^{O(1)})$. Here q and t are the parameters used in the locks.

In each iteration of the protocol, The verifier queries an entry of $\mathsf{MIX}(r')$, which is of length $O(qt) + O(\log n + \log k + \log m)$. It then checks $O(1)$ locks requiring $O(1)$ probes each.

Soundness: We argue that if the original PCP had error at most $1-s$, i.e., the verifier accepts $x \notin L$ with probability at most $1-s$, then the new PCP will have error at most $1-s/16$. First recall that a lock may be opened at most one way without incurring a $1/16$ probability of rejection. By inspection of the protocol, there is never any advantage to constructing a lock that can't be opened, so we assume without loss of generality that every lock has a well-defined content. Let P be the PCP obtained by taking the contents of PCP's locks, and define $\pi(r)$ as the contents of $\mathsf{PERM}(r)$ (π is not necessarily a permutation).

Denote by $V[r]$ the run of verifier V using random coin tosses r. We say that $r' \in R$ is good if there exists an $r \in R$ such that $r' = \pi(r)$ and $V_0[r]$ accepts P, and bad otherwise. There are at least $s|R|$ bad r', since V_0 accepts on at most $(1-s)|R|$ tapes. We now argue that if V chooses a bad r' he will reject with probability at least $1/16$; this will complete the soundness analysis. Let $\mathsf{MIX}(r') = (r, K_a, K_b, K_c, K_r)$. If $\pi(r) \neq r'$ then V will reject with probability at least $1/16$ in Step 2. If $\pi(r) = r'$ then $V_0[r]$ must reject P (or r' would be good), in which case V will either reject with probability at least $1/16$ in Step 3 (if K_a, K_b or K_c try to unlock incorrect values) or reject with probability 1 in Step 4 (since $V[r]$ rejects P).

The soundness of the PCP may be amplified by repetitions in the standard manner.

Robustness: We give only a brief sketch of this analysis. We first argue that if (a possibly malicious) V makes less than $\min(m, U)$ queries to ($\mathsf{PCP}, \mathsf{PERM}, \mathsf{MIX}$) then, unless a "bad event" happens, V's view can be simulated statistically closely. Here, U $(= \sqrt{2^t/n})$ is a parameter of the lock box. We then show that such bad events occur with negligible probability.

First, we simplify matters in a manner that only helps the adversary. If V queries any bit of $\mathsf{MIX}(i)$ then we give it the entire contents of $\mathsf{MIX}(i) = (r, K_a, K_b, K_c, K_r)$, the values for the original PCP at locations a, b and c (i.e. b_a, b_b and b_c) and the values of the locks L_a, L_b, L_c and L_r. We assume that V queries $\mathsf{MIX}(i)$ only once for each i.

Our simulator works as follows. Whenever V queries $\mathsf{MIX}(r')$, the simulator S performs the following steps:

1. S randomly chooses $r \in R$, r not previously chosen, and sets $\pi(r) = r'$.
2. S computes a, b and c, the indices queried by $V_0[r]$
3. S invokes the simulator for the original PCP to obtain simulated b_a, b_b and b_c. Note that this simulation depends on the values of b_i that may already have been specified; indeed, some or all of (b_a, b_b, b_c) may already have been determined.
4. S invokes the locker simulator to produce
$$(K_a, L_a), (K_b, L_b), (K_c, L_c), \text{ and } (K_r, L_r).$$
This simulation is performed based on the now determined b_a, b_b, b_c and $\pi(r)$ and on the previously determined (by the simulation) bits of L_a, L_b, L_c and L_r.

Whenever V queries bits of $\mathsf{PERM}(r) = L_r$ or $\mathsf{PCP}(i) = L_i$, S calls the lock simulator to simulate the values of these bits. In some cases, L_r or L_i may have been entirely determined by the simulation, due to the simulation of a $\mathsf{MIX}(r')$ query.

If the simulators for the locking system and the original PCP were always statistically close, then by a straightforward argument the resulting simulation would be statistically close. However, sometimes the lower-level simulations would abort, detecting a situation in which they could no longer guarantee a close simulation. We now bound the probability of such bad events. First, the simulation of the original PCP is performed essentially independent from the other simulations. We first note that whenever it is invoked it is on a random string r that is new but otherwise random. By Lemma 2.1, up to m simulations can be performed with a bad event occuring with probability at most $1/m^k$.

Similarly, the simulator for each locker can handle up to U queries, and the difference between the output of the simulation and the view obtained by using actual lock boxes will be less than $1/n^c$ for any c, as n grows sufficiently large. Since at most U simulations are invoked, the difference will be at most U/n^c.

4.3 Setting the Parameters

To prove Theorems 1 and 2, we give appropriate settings for the parameters m and k used in Lemma 2.1 and for t and q used in the locker construction. Recall that n is the size of the original PCP (though we may artificially make it larger, as discussed below). For Theorem 1, we can set $k = t = q = c \log n$ for c a sufficiently large constant, so that $\sqrt{2^t/n} \geq U$. We set m to be at least U. Note that m^{-k} is negligible. Also, note that for large c, as n grows sufficiently large, the deviations due to the lock box simulators (U/n^c) will be less than $n^{c_0 - c}$ as n grows sufficiently large.

Note that we are being archaic in our notion of security, typically one allows for a security parameter so greater security can be obtained for the proof of a fixed-sized statment. We can obtain the same effect by deliberately padding n, the size of the original PCP. Details are omitted.

For Theorem 2 we assume without loss of generality that the original PCP proving L has size $n = 2^{|x|^{c_1}}$ for c_1 sufficiently large (if not, we can pad the PCP). We then can set $U = 2^{|x|}$, and set $k = t = q = |x|^c$, for c large enough so that $\sqrt{2^t/n} \geq U$. Details are omitted.

5 Acknowledgments

We thank Sanjeev Arora, Noam Nisan, and Mike Saks for helpful discussions.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 13–22, 1992.

[2] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions, Proc. of STOC88.

[3] C. Dwork, U. Feige, J. Kilian, M. Naor, S. Safra, "Low Communication, 2-Prover Zero-Knowledge Proofs for NP" *Advances in Cryptology: Crypto '92*, pages 217–229.

[4] U. Feige and J. Kilian. "Zero-Knowledge and the Chromatic Number" In *Proceedings, 1996 Conference on Computational Complexity.*

[5] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, 18 (1):186–208, 1989.

[6] J. Håstad. Testing of the long code and hardness for clique *Proc. 28th ACM Symp. on Theory of Computing*, pages 11–19,1996.

[7] J. Kilian. Uses of randomness in algorithms and protocols. Ph.D. thesis.

[8] J. Kilian and M. Naor. On the Complexity of Statistical Reasoning. In *Proceedings, Israeli Symposium on Theory of Computing and Systems*, pages 209–217, 1995.