On Limitations of Universal Simulation: Constant-Round Public-Coin Zero-Knowledge Proofs Imply Understanding Programs

Yi Deng*, Juan Garay†, San Ling‡, Huaxiong Wang‡ and Moti Yung♮

* Institute of Information Engineering, Chinese Academy of Sciences, China
† AT&T Labs – Research, USA
‡ School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
♮ Google Inc., USA

September 10, 2013

Abstract. In this paper we consider the problem of whether there exist non-trivial constant-round public-coin zero-knowledge (ZK) proofs. We focus on the type of ZK proofs that admit a universal simulator (which handles all malicious verifiers), and show a connection between the existence of such proof systems and a seemingly unrelated "program understanding" problem: for a natural class of constant-round public-coin ZK proofs (which we call "canonical," since all known ZK protocols fall into this category), a universal simulator can actually be used (as an oracle) to distinguish a non-trivial property of the verifier's program.

Our result can be viewed as new and extended evidence against the existence of constant-round public-coin ZK proofs, since the above program-understanding problem, a typical goal in reverse-engineering attempts, is commonly believed to be notoriously hard—in general, and particularly so in the case of limited straight-line simulators. The earlier negative evidence for the above is Barack, Lindell and Vadhan [FOCS '03]'s result, which was based on the incomparable assumption of the existence of certain entropy-preserving hash functions, now known not to be achievable from standard assumptions.

Our reduction combines a careful analysis of the behavior of a set of verifiers in the above protocols and simulation, with a key tool which is an improved structure-preserving version of the well-known Babai-Moran Speedup (de-randomization) Theorem.

1 Introduction

In their seminal paper [23], Goldwasser, Micali and Rackoff introduced the fascinating notion of a *zero-knowledge* (ZK) interactive proof, in which a party (called the prover) wishes to convince another party (called the verifier) of some statement, in such a way that the following two properties are satisfied: (1) zero knowledge— the prover does not leak any knowledge beyond the truth of the statement being proven, and (2) soundness—no cheating prover can convince the verifier of a false statement except with small probability. A vast amount of work ensued this pioneering result. Shortly after the introduction of a ZK proof, Brassard, Chaum and Crépeau [3] defined a ZK proof system with relaxed soundness requirement, called a ZK *argument*, for which soundness is only required to hold against polynomial-time cheating provers.

The original ZK proof system for the quadratic residuosity problem presented in [23] is of a special form, in which the verifier simply sends independently random coins at each of his steps. Such a proof system is called a *public-coin* proof system, and has been found to be more broadly applicable and versatile than "private-coin" proof systems. Another notable feature of this type of proof systems is its round efficiency, as it consists of only 3 rounds, i.e., just 3 messages are exchanged in a session. This round efficiency, however, brings about a side effect of soundness error, which is too large to be used in cryptographic settings where typically a negligibly small soundness error is required. Indeed, there seems to be a tradeoff between round efficiency and soundness error for public-coin proof system: we can achieve negligible soundness error by sequential repetition, but then the resulting system is no longer constant-round. This is in contrast with private-coin ZK proof systems, for which constant rounds and negligible soundness error can be achieved simultaneously.

In fact, whether constant-round public-coin ZK protocols (or even argument systems) with negligible soundness error exist for some non-trivial language has been a long-standing open problem. In [21], Goldreich and Krawczyk showed that, for non-trivial languages, the zero knowledge property of such a proof system cannot be proven via black-box simulation. Black-box simulation was in fact the only known technique to demonstrate "zero-knowledgeness" for a long while, and hence the Goldreich-Krawczyk result was viewed as strong negative evidence against the existence of constant-round public-coin ZK proof systems.

A breakthrough result in 2001 changed the state of things. Indeed, in [2] Barak presented a non-black-box ZK argument in which the simulator makes use of the code of the malicious verifier in computing the prover messages (albeit without understanding it). Barak's construction follows the so-called "FLS paradigm" [19], which consists of two stages. In the first stage the prover sends a commitment c to a hash value of an arbitrary string, to which the verifier responds with a random string r; in the second stage, the prover proves using a witness indistinguishable (WI) universal argument that either the statement in question is true or c is a commitment to a hash value of some code Π, and, given input c, Π outputs r in some super-polynomial time. Note that this is a constant-round public-coin argument, and that its simulator does not "rewind" the malicious verifier (and it is hence called a *straight-line* simulator) and, furthermore, runs in strict polynomial time. These features have been proved impossible to achieve when using black-box simulation [21,7].

Barak's argument system still left open the question whether non-trivial constant-round public-coin (non-black-box) ZK *proof* systems exist. *Prima facie*, being able to extend his technique to a proof system seems challenging, mainly due to the fact that since a Turing machine or algorithm may have an arbitrarily long representation, a computationally unbounded prover may, after receiving the second verifier message r, be able to find a program Π (whose description may be different from the verifier's with which the prover is interacting) such that, $c = \mathrm{Com}(h(\Pi))$, and on input c, Π outputs r in the right amount of time.

In [9], Barak, Lindell and Vadhan showed further negative evidence for the above problem, by proving that if a certain class of entropy-preserving hash functions exist, then such a proof system cannot exist. Their formulation of entropy-preserving hash functions is mathematically simple, inspiring further research to base such hash functions on standard assumptions. Unfortunately, we do not have a candidate for such functions thus far, and furthermore, as showed in recent work by Bitansky *et al.* [4], such functions *cannot* be based on any standard assumption via black-box reduction.

Our results and techniques. In this paper, we provide evidence of a different nature against the existence of constant-round public-coin ZK proof systems. We focus on the type of ZK proofs that admit a universal simulator, i.e., ZK proof systems for which there is a single simulator that can handle all malicious verifiers. To our knowledge, all constructions of ZK proofs in the literature are of this type.

We uncover an unexpected connection between the existence of such proof systems and a seemingly unrelated "program-understanding" problem: for a natural class of constant-round public-coin ZK proofs (which we call "canonical," as all known ZK protocols fall in this category), a universal simulator for such ZK proof system can actually be used to figure out some non-trivial property of a verifier's program functionality. More specifically, we show that, given a constant-round public-coin ZK proof system $\langle P, V\rangle$, there exist a step index k and a set of polynomial number of verifiers that share the verifier next-message functions up to the $(k-1)$-th step but have t distinct k-th next-message functions, for t a polynomial, denoted by $(V_k^1, V_k^2, ..., V_k^t)$, such that for any polynomial-time constructible code V_k^* that is promised to have the same functionality as one of V_k^i's in the above set, the universal simulator, taking V_k^* as input, can generate a session prefix before the k-th verifier step that enables us to single out a V_k^j in the set which is functionally *different* from V_k^*.

Our result can be viewed as strong negative evidence on the existence of constant-round public-coin ZK proof systems. On one hand, devising a rewinding technique that could be used in the simulation of such a proof appears fairly unfathomable; on the other hand, if such a proof does admit a straight-line simulator, then the above result shows that we would be able to figure out some non-trivial functionality/property of V_k^* *without executing it* (since in producing the session prefix before the k-th verifier step, the simulator does not run V_k^* at all!), which seems to be extremely unlikely.

One key tool in our reduction is an improved structure-preserving version of the well-known Babai-Moran Speedup (derandomization) Theorem [1,10,11], which essentially says that, for a constant-round public-coin interactive proof system in which the verifier sends m messages and each of the prover messages is of length p, if the cheating probability for an unbounded prover is ϵ, then there exist $(p/O(\log \frac{1}{\epsilon}))^m$ verifier random tapes such that the cheating probability for the unbounded prover over these tapes is bounded away from 1—and this holds even when the prover knows this small set of random tapes in advance. (In our setting, the original Babai-Moran theorem yields a much larger size $((O(p))^m)$ of such set of verifier random tapes.) In addition, we show that this is tight with respect to round complexity, in the sense that there are public-coin proof systems with a super-constant number of rounds for which the prover's cheating probability is 1, over any polynomial number of verifier random tapes.

Related work. As mentioned above, Barak, Lindell and Vadhan [9] conjectured the existence of certain entropy-preserving hash functions and proved that the conjecture's veracity would rule out the possibility of existence of constant-round public-coin ZK proof systems. Recent work by Bitansky *et al.* [4], however, showed that this conjecture cannot have a black-box reduction from any standard assumption. Our result is incomparable to [9]'s in the following sense: while our result refers to a somewhat more restricted type of constant-round public-coin ZK proofs, our underlying assumption, the hardness of program understanding, appears to be more solid than the existence of entropy-preserving hash functions on which the negative result of Barak, Lindell and Vadhan is based.

A somewhat related problem to our understanding problem is program obfuscation, the theoretical study of which was initiated by Barak *et al.* [6]. At a high level, an obfuscator is an efficient compiler that takes a program as input and outputs an "unreadable" program with the same functionality as the input program. Obfuscation has recently attracted a lot of research efforts (e.g., [12,25,27,30]) due to its wide range of applications, from software protection to providing a justification to the random oracle model. Hada [26], in particular, showed that the existence of a certain type of ZK protocol is tightly related to the existence of an obfuscator for some specific functionality. Unfortunately, for a large class of functionalities, it has been shown that obfuscators do not exist.

We stress that the impossibility of code obfuscation does not imply that understanding the functionality of a given code is tractable. Breaking the security of an obfuscator for some specific functions seems to at least require execution of the target obfuscated code; thus, if a constant-round public-coin proof system admits a straight-line simulator as the one for Barak's ZK argument, then our reduction guarantees an algorithm that could solve our program-understanding problem *without* any execution of the target code, a problem which appears to be (much) harder than breaking obfuscators.

Organization of the paper. Preliminaries, notation and definitions that are used throughout the paper are presented in Section 2. Definitions of *canonical* ZK proofs and of the *verifier-understanding* problem are formulated in Section 3. The improved derandomization lemma and the reduction of constant-round public-coin ZK proofs to the verifier-understanding problem are presented in Section 4. For the sake of readability, some of the proofs presented in the main body are only sketches; the full proofs can be found in the appendix.

2 Preliminaries

In this section we recall some definitions and introduce notation that will be used throughout the paper.

We say that function $\mathsf{neg}(n)$ is *negligible* if for every polynomial $q(n)$ there exists an N such that for all $n \geq N$, $\mathsf{neg}(n) \leq 1/q(n)$. Throughout this paper, polynomials always refer to polynomials in the security parameter n of a proof system.

When referring to a Turing machine M, we will slightly abuse notation and use M to represent both its code and its functionality. Specifically, if we write $M \in \mathcal{G}$ for some set $\mathcal{G}$, we will mean that there is a Turing machine in $\mathcal{G}$ whose code is identical to the code of M; on the other hand, if we say that M^* is "functionally equivalent" to M (as defined below), both M^* and M will clearly refer to their functionality.

We think of an interactive Turing machine as a machine that computes a collection of next-message functions. (We refer the reader to [20] for a rigorous definition.)

Definition 1. *For two deterministic (interactive) Turing machines M^1 and M^2, we say M^1 and M^2* have the same functionality*, or* are functionally equivalent *if they compute the same collection of next-message functions. That is, for any input* hist*, the next message produced by M^1 is identical to the one produced by M^2—i.e., $M^1(\mathsf{hist}) = M^2(\mathsf{hist})$.*

We will use $M^1 \stackrel{\mathrm{f}}{=} M^2$ as a shorthand for the above, and $M^1 \stackrel{\mathrm{f}}{\neq} M^2$ as its negation.

An *interactive proof system* $\langle P, V\rangle$ for a language L is a pair of interactive Turing machines in which the prover P wishes to convince the verifier V of some statement $x \in L$. In an interaction between P and V, the *view* of V, denoted by View_V^P, consists of the common input x,V's random tape, and all the prover messages it received. The *round complexity* of an interactive proof system $\langle P, V\rangle$ is the number of messages exchanged in an execution of $\langle P, V\rangle$. Without loss of generality, in this paper we assume that the verifier V sends the first message; thus, if the verifier sends m messages in total, the round complexity of this proof system is $2m$.

Definition 2 (Interactive Proofs). *A pair of interactive Turing machines $\langle P, V\rangle$ is called an* interactive proof system for language L *if V is a probabilistic polynomial-time (PPT) machine and the following conditions hold:*

- COMPLETENESS: *For every $x \in L$,* $\Pr[\langle P, V\rangle(x) = 1] = 1$.
- SOUNDNESS: *For every $x \notin L$, and every (unbounded) prover P^*,* $\Pr[\langle P^*, V\rangle(x) = 1] < \mathsf{neg}(|x|)$.

Public-coin proof systems and verifier decomposition. An interactive proof system is called *public-coin* if at every verifier step, the verifier sends only truly random messages.

We will use boldface lowercase letters to refer to the verifier's random tapes (e.g., $\mathbf{r}$), and italic for each verifier message (e.g., r). Thus, for a $2m$-round public-coin interactive proof system $\langle P, V\rangle$, we have $\mathbf{r} = [r_1, r_2, ..., r_m]$, where r_i is the i-th verifier message. We use superscripts to distinguish different verifier's random tapes; e.g., $\mathbf{r}^i$, $\mathbf{r}^j$, etc.

Given a random tape $\mathbf{r} = [r_1, r_2, ..., r_m]$, we can "decompose" the verifier $V(\mathbf{r})$ into a collection of next-message functions, $V = [V_1, V_2, ..., V_m]$, with each V_i being defined as:

$$r_i \text{ or } \perp \leftarrow\ V_i(\mathsf{hist}, r_1, r_2, ..., r_i),$$

where hist refers to the current history up to the $(i-1)$-st prover step ; that is, given hist, $V_i(\mathsf{hist}, r_1, r_2, ..., r_i)$ outputs r_i if hist is accepting, or aborts if not. Note that the next message function V_i needs the randomness $[r_1, r_2, ..., r_{i-1}]$ of previous verifier steps in order to check whether the current history is accepting or not.

We will sometimes abbreviate and use superscripts to distinguish verifiers running on different random tapes; that is, given two random tapes $\mathbf{r}^i = [r_1^i, r_2^i, ..., r_m^i]$ and $\mathbf{r}^j = [r_1^j, r_2^j, ..., r_m^j]$, we will use V^i and V^j as a shorthand for $V(\mathbf{r}^i)$ and $V(\mathbf{r}^j)$, respectively. Similarly, we will use V_k^i to denote the k-th next-message function of the verifier $V(\mathbf{r}^i)$.

Now, given a verifier $V^i = [V_1^i, ..., V_m^i]$, we will use $V_{[j,k]}^i$ to denote the partial verifier strategy starting with the j-th next message function and up to the k-th next message function. We will typically be concerned with the following partial strategies:

$$\text{prefix strategy: } V_{[1,k]}^i \triangleq [V_1^i, V_2^i, ..., V_k^i];$$
$$\text{suffix strategy: } V_{[k,m]}^i \triangleq [V_k^i, V_{k+1}^i, ..., V_m^i].$$

A real-world-interaction version of simulation and universally simulatable ZK proofs. We first present the standard definition of ZK proofs.

Definition 3 (Zero-Knowledge Proofs). *An interactive proof system $\langle P, V\rangle$ for a language L is said to be* universally simulatable zero-knowledge *if for any probabilistic polynomial-time V^* and any $x \in L$, there exists a probabilistic polynomial-time algorithm S such that the distribution $\{\mathrm{View}_{V^*}^{P}\}_{x\in L}$ is computationally indistinguishable from the distribution $\{S(x, V^*)\}_{x\in L}$.*

The standard simulation process for a malicious verifier V^* is typically as follows. The PPT simulator S, taking the common input x and V^*'s code as inputs, is to output a session transcript. S treats V^* as a subroutine, interacting (with possible "rewinds") with it *internally*, and outputting a view of V^* as the result of the interaction. Without loss of generality, one can think of the output of the simulator as the final (internal) interaction between $S(x, V^*)$ and V^*.

In this paper, we wish to treat S as an oracle and be able to obtain prover messages from S one by one, rather than obtaining the entire session transcript at once. For this purpose, we make the above (final) internal interaction "external," by casting the simulation process for a malicious verifier V^* as a real interaction between $S(x, V^*)$ (playing the role of the prover) and an external V^*, and whenever S wants to rewind V^*, it does it on its own copy of V^*. We denote this interaction by $(S(x, V^*) \Leftrightarrow V^*)$, and the view of V^* resulting from this interaction by $\{\mathrm{View}_{V^*}^{S(x,V^*)}\}_{x\in L}$. (For brevity, we will sometimes drop x from the above notation.)

The following fact is easy to verify.

Fact 1. For any x and any V, V^* such that $V \stackrel{\mathrm{f}}{=} V^*$, $(S(x, V^*) \Leftrightarrow V^*)$ generates the same session transcript as $(S(x, V^*) \Leftrightarrow V)$.

We conclude this section with the following definition of *universally simulatable ZK proof*, which differs from the standard ZK definition in the order of quantifiers ("$\exists S \forall V^*$" instead of "$\forall V^* \exists S$").[1]

Definition 4 (Universally Simulatable Zero-Knowledge Proofs). *An interactive proof system $\langle P, V\rangle$ for a language L is said to be* universally simulatable zero-knowledge *if there exists a probabilistic polynomial-time algorithm S such that for any probabilistic polynomial-time V^* and any $x \in L$, the distribution $\{\mathrm{View}_{V^*}^{P}\}_{x\in L}$ is computationally indistinguishable from the distribution $\{\mathrm{View}_{V^*}^{S(V^*)}\}_{x\in L}$.*

[1] To our knowledge, all known ZK proofs are actually universally simulatable, satisfying this stronger requirement.

3 Canonical ZK Proofs and the *Verifier-Understanding* Problem

In this paper we will focus on ZK proof systems with a certain property, which we call "canonical," since all known constructions (see below) fall in this category. We first give some intuition behind it. We observe that for many ZK protocols, if the simulation is formulated as an interaction between $S(V^*)$ and V^*, as in the previous section, then for a successful simulation to take place it is sufficient to feed S with only *partial* code of V^*, rather than with its entire code.

Illustrative examples are those ZK protocols following the popular FLS paradigm [19]. Recall that in this paradigm, a ZK protocol consists of two stages: in the first stage, the prover and the verifier set up a trapdoor (which is useful for the simulation), and then, in the second stage, the prover proves that either the statement being proven is true or that he knows the trapdoor in a WI protocol. Hence, it is easy to see that if the code V^* of a malicious verifier is given by two separate specifications V_{I}^* and V_{II}^*, representing the first and second stages of V^*, respectively, then the simulator can perform a successful simulation given only V_{I}^*, since it can extract the trapdoor from it, which, no matter what V_{II}^* is, it enables it to simulate the second stage in a straight-line fashion. That is, using the notation from last section, for any second-stage honest verifier V_{II} (which may have a different functionality from V_{II}^*'s), both interactions $(S(V^*) \Leftrightarrow [V_{\mathrm{I}}^*, V_{\mathrm{II}}])$ and $(S(V_{\mathrm{I}}^*) \Leftrightarrow [V_{\mathrm{I}}^*, V_{\mathrm{II}}])$ are accepting. This, in a nutshell, is what the canonical property says—if the former interaction is accepting for any V_{II}, so is the second interaction.

Its formal definition makes use of the following definition about session prefixes of proof systems.

Definition 5 **(*Good/bad* session prefix).** *Let $\langle P, V\rangle$ be a $2m$-round public-coin proof system for a language L, and let $\mathcal{V}_{[1,\ell]}$ denote the set of verifiers that share the same verifier prefix strategy $V_{[1,\ell]}$, for some $1 \leq \ell \leq m$. We call a session prefix $(r_1, p_1, ..., p_\ell)$* good *with respect to $\mathcal{V}_{[1,\ell]}$ if there is a residual (unbounded) prover strategy with auxiliary input $\mathcal{V}_{[1,\ell]}$ which, based on this session prefix, can make a verifier randomly chosen from $\mathcal{V}_{[1,\ell]}$ accept with probability 1. Otherwise, we call it a* bad *session prefix with respect to $\mathcal{V}_{[1,\ell]}$.*

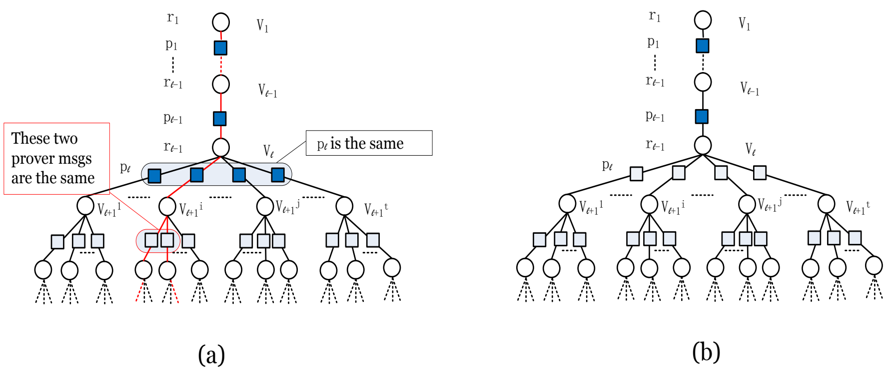

Fig. 1. A *good* session prefix (a) and its robustness (b). Each node (circle) represents a verifier next-message function, or equivalently (in our case of public-coin proof systems), a random string that is used in this step. Each path represents a (complete) interaction with an honest verifier.

Equivalently, we call a session prefix $(r_1, p_1, ..., p_\ell)$ "good" with respect to $\mathcal{V}_{[1,\ell]}$ if the following holds, which can be decided in time exponential in the length of the prover's messages. Let poly be the size of $\mathcal{V}_{[1,\ell]}$. Then there are poly number of session continuations of the form $(r_{\ell+1}, ..., p_m)$, each assigned to a verifier in $\mathcal{V}_{[1,\ell]}$, such that the following conditions hold:

1. Every verifier in $\mathcal{V}_{[1,\ell]}$ will accept the transcript $(r_1, p_1..., p_\ell, r_{\ell+1}, ..., p_m)$ assigned to it.
2. If two verifiers in $\mathcal{V}_{[1,\ell]}$ share the same prefix strategy up to the ℓ'-th step, $\ell \le \ell' \le m$, then the two transcripts assigned to them share the same session prefix $(r_1, p_1..., r_{\ell'}, p_{\ell'})$.

A good session prefix is pictorially depicted in Figure 1(a). In the figure, if $(r_1, p_1, ...p_\ell)$ is good with respect to the tree, then for every edge below V_ℓ, we can assign a prover message to it such that: (1) each path is accepting, and (2) for every two paths that share the same prefix strategy up to the ℓ'-th verifier step, $\ell \le \ell' \le m$ (e.g., the red paths), the session prefixes of these two paths up to the ℓ'-th prover step are the same.

In addition, one can easily verify the following "robustness" fact about a good session prefix: if a session prefix $(r_1, p_1, ..., p_\ell)$ is good with respect to $\mathcal{V}_{[1,\ell]}$, then for any $1 \le i \le \ell$, the session prefix $(r_1, p_1, ..., p_i)$ is also good with respect to $\mathcal{V}_{[1,\ell]}$. See Figure 1(b). The figure illustrates the fact that if $(r_1, p_1, ..., p_{\ell-1}, r_\ell, p_\ell)$ is good, so is the (sub)prefix $(r_1, p_1, ..., p_{\ell-1})$ with respect to the same tree. This is because all prover messages on edges below V_ℓ (including p_ℓ) simply satisfy the two conditions that make a session prefix good.

We are now ready to define what we call *canonical* ZK proofs (note that they are defined conditionally).

Definition 6 (*Canonical* ZK Proofs). *Let $\langle P, V\rangle$ be a $2m$-round universally simulatable ZK proof system for a language L (Definition 4), S be the associated simulator and t be some polynomial. We call $\langle P, V\rangle$* canonical *if for any common input x (not necessarily in L), every set $\mathcal{V}_{[1,k-1]}$ of verifiers that share prefix strategy $V_{[1,k-1]}$, $2 \le k \le m$ (cf. Definition 5), but with t distinct k-th step strategies $V_k^1, V_k^2, ..., V_k^t$, the following holds.*

For any verifier code $V^*_{[1,k-1]}$ satisfying $V^*_{[1,k-1]} \overset{\mathrm{f}}{=} V_{[1,k-1]}$, if, for some $1 \le i \le t$, there exists $V_k^* \overset{\mathrm{f}}{=} V_k^i$ such that the session prefix $(r_1, p_1, ..., p_{k-1}) \leftarrow (S([V^*_{[1,k-1]}, V_k^*]) \Leftrightarrow [V^*_{[1,k-1]}, V_k^*])$ is good with respect to $\mathcal{V}_{[1,k-1]}$, then S, taking only $V^*_{[1,k-1]}$ as input, can also produce a session prefix (i.e., $(r_1, p'_1, ..., p'_{k-1}) \leftarrow (S(V^*_{[1,k-1]}) \Leftrightarrow V^*_{[1,k-1]})$) which is good with respect to $\mathcal{V}_{[1,k-1]}$.

A canonical ZK proof is depicted in Figure 2.

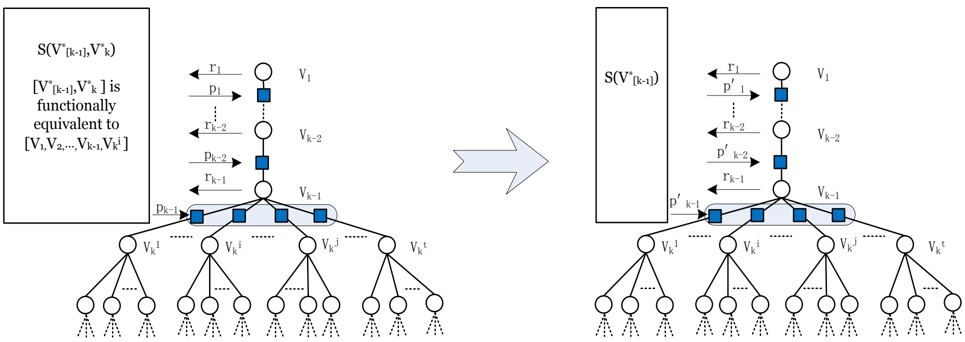

Fig. 2. A canonical ZK proof.

Remark. We stress that, logically, the property above makes a restriction *only* on the type of ZK proofs that satisfy the "if condition" in its definition, and does not require the simulator with partial code of a verifier ($S([V^*_{[1,k-1]}, V_k^*])$) to generate a valid session prefix. The property states that if this happens and the session prefix is good, then S can do the same *without* being given verifier code V_k^*. To our knowledge, all constructions of ZK protocols enjoy this property—cf. the FLS example at the beginning of the section, as well as those protocols that do not follow FLS paradigm, such as, for example, Blum's 3-round ZK proof for Graph Hamiltonicity [8] (which does not satisfy the "if condition" of the canonical property, and thus falls in the type of canonical ZK proofs.).

We are now ready to formulate the "verifier-understanding problem," to which the existence of constant-round public-coin ZK proofs is reduced. In a nutshell, given a set of distinct verifier k-th next-message-functions, the problem resides in constructing an "understanding" algorithm U, with oracle access to simulator S, such that for any polynomial-time constructible program V_k^* that is promised to be functionally equivalent to one of the next-message functions, is able to discern one from the set that is functionally *different* from V_k^*. Formally:

Definition 7 (The *Verifier-Understanding* Problem). *Let $\langle P, V\rangle$ be a $2m$-round canonical ZK proof system for a language L (Definition 6), S be its simulator, p the length of each prover's message, and t a polynomial in the security parameter n. Given are a set $\mathcal{V}_{[1,k-1]}$ of deterministic honest verifiers that share the same prefix verifier $V_{[1,k-1]}$, but have t* distinct *k-th next-message functions $V_k^1, V_k^2, ..., V_k^t$, denoted by set $\mathcal{V}_k$, and an auxiliary input* aux[2]. *The* verifier-understanding problem *is to find a non-uniform algorithm U, running in time $2^{O(p)}$, such that for every polynomial-time algorithm C, the following holds:*

- *First, C picks a machine $V_k^i \in \mathcal{V}_k$ at random and outputs a polynomial-time Turing machine V_k^* such that $V_k^* \overset{\mathrm{f}}{=} V_k^i$.*
- *Next, U, making a constant number of queries to the oracle $S(\mathsf{aux}, V_k^*)$, outputs $V_k^j \in \mathcal{V}_k$ such that $V_k^j \overset{\mathrm{f}}{\neq} V_k^*$ with probability negligibly close to 1. I.e.,*

$$\Pr\left[V_k^* \leftarrow C(V_k^1, V_k^2, ..., V_k^t, i);\ j \leftarrow U^{S(\mathsf{aux}, V_k^*)}(\mathcal{V}_{[1,k-1]}, \mathcal{V}_k) : V_k^* \overset{\mathrm{f}}{\neq} V_k^j\right] > 1 - \mathsf{neg}(n),$$

where the probability is taken over the random choice i and the randomness used by C,U and S.

We stress that in the above definition, algorithm U is not given V^*'s code as input. This captures U's difficulty in understanding what the program does, specially when its oracle S *does not* execute V^*.

4 Constant-Round Public-Coin Zero-Knowledge Proofs Imply Understanding Programs

We are now ready to present our main result, which exhibits a reduction from constant-round public-coin canonical ZK proofs to the verifier-understanding problem (Definition 7), a problem seemingly quite different in nature. We first fix some parameters and revisit notation:

- $\langle P, V\rangle$: A $2m$-round public-coin canonical ZK proof sytem for some constant m. We let n be the security parameter and p be the length of each prover's message.
- $\mathcal{V}_{[1,k-1]}$: A set of deterministic honest verifiers that share the same (honest) prefix verifier $V_{[1,k-1]}$, but have t *distinct* k-th step functions $V_k^1, V_k^2, ..., V_k^t$; $|\mathcal{V}_{[1,k-1]}| \le q$, where t and q are polynomials (defined in Lemma 1)[3].
- $\mathcal{V}_k$: The set $\{V_k^1, V_k^2, ..., V_k^t\}$, as above.
- $V'_{[1,k-1]}$: The auxiliary input to S, which is the code of a prefix verifier such that $V'_{[1,k-1]} \overset{\mathrm{f}}{=} V_{[1,k-1]}$. (When $k = 1$, it is set to the empty string.)

[2] This auxiliary input is given to S; in our main theorem (Theorem 1) it will be the code of some verifier prefix strategy.

[3] At the k-th verifier step, the number of distinct next-message functions should in fact be t_k. For simplicity, we assume $t = t_k$ for all $1 \le k \le m$.

We now show that if $\langle P, V \rangle$ admits an universal simulator S, then there is an algorithm U^S, taking $\mathcal{V}_{[1,k-1]}$ and $\mathcal{V}_k$ as auxiliary inputs, which can solve the verifier-understanding problem (cf. Definition 7) with respect to verifier set $\mathcal{V}_k$. Formally:

Theorem 1. *Let $\langle P, V \rangle$ be a $2m$-round, public-coin canonical ZK proof system with negligible soundness error for a non-trivial language $L \notin \mathcal{BPP}$, and S be its universal simulator. Then, there exist $x \notin L$, a constant k, $2 \le k \le m$, sets $\mathcal{V}_{[1,k-1]}$ and $\mathcal{V}_k$, a verifier code $V'_{[1,k-1]}$ as above, and an algorithm U, making only $k-1$ queries to $S(\cdot)$ and running in time $2^{O(p)}$, such that, for any polynomial-time algorithm C that on input $(\mathcal{V}_k, i)$, $1 \le i \le t$ outputs V_k^* satisfying $V_k^* \overset{\mathrm{f}}{=} V_k^i \in \mathcal{V}_k$, the following holds:*

$$\Pr\left[V_k^* \leftarrow C(\mathcal{V}_k, i); j \leftarrow U^{S(V'_{[1,k-1]}, V_k^*)}(\mathcal{V}_{[1,k-1]}, \mathcal{V}_k) : V_k^* \overset{\mathrm{f}}{\neq} V_k^j \in \mathcal{V}_k\right] > 1 - \mathsf{neg}(n),$$

where the probability is taken over the random choice i and the randomness used by C, U and S.

We now give a high-level sketch of proof of the theorem, which mainly consists of three steps. (Refer to Figure 3.)

1. We first prove a de-randomization lemma that can be viewed as a structure-preserving version of the Babai-Moran "Speedup Theorem" [10] (with improved parameters), which essentially says that for a constant-round public-coin interactive proof systems $\langle P, V \rangle$ for some non-trivial language in which the verifier sends m messages and each of the prover's messages is of length p, if the cheating probability for an unbounded prover is negligible, then there exists a polynomial q number of random tapes for the verifier such that the cheating probability for the unbounded prover over these verifier's random tapes is less than $1-1/q$. Denote these q deterministic verifiers by $V^1, V^2, ..., V^q$. The various trees in Figure 3(a) correspond to these q verifiers.
2. Next, we show that there exists a false statement x such that for every verifier V^i, $1 \le i \le q$, and any polynomial-time constructible code V^* which is functionally equivalent to V^i, the session $(S(V^*) \Leftrightarrow V^*)$ (which, by Fact 1 is identical to $(S(V^*) \Leftrightarrow V^i)$) is accepting except with negligible probability. This is shown in Figure 3(a).
3. Finally, we prove that among these q verifiers, we can find a (sub)tree $\mathcal{V}_{[1,k-1]}$ that has the same prefix strategy $[V_1, V_2, ..., V_{k-1}]$ up to the $(k-1)$-th verifier step but "splits" at the k-th verifier step, and a code $V'_{[1,k-1]}$ that is functionally equivalent to $V_{[1,k-1]} = [V_1, V_2, ..., V_{k-1}]$, such that, for any polynomial-time constructible code V_k^* that is promised to be functionally equivalent to one of those V_k^i's (nodes) at level k, the following two conditions hold:
   - The session prefix $(r_1, p_1, ..., p_{k-1})$ produced by $(S([V'_{[1,k-1]}, V_k^*]) \Leftrightarrow V'_{[1,k-1]})$ (or equivalently, by $(S([V'_{[1,k-1]}, V_k^*]) \Leftrightarrow V_{[1,k-1]}))$ is *bad* with respect to $\mathcal{V}_{[1,k-1]}$ (cf. Definition 5). This implies that there is a subtree (e.g., the one in the red underbrace in Figure 3(b)) in $\mathcal{V}_{[1,k-1]}$, with respect to which $(r_1, p_1, ..., p_{k-1})$ is bad.
   - However, the session prefix $(r_1, p_1, ..., p_{k-1})$ is *good* with respect to the subtree that shares the same prefix strategy $[V_{[1,k-1]}, V_k^i]$ (the one in the blue underbrace in Figure 3(b)).

   This enables us to construct an algorithm (with oracle access to S) that is able to "understand" the code V_k^*, by pin-pointing another verifier code, say, V_k^j, such that $V_k^j \overset{\mathrm{f}}{\neq} V_k^*$.

We now present our new de-randomization lemma, followed by the remaining details of the proof of Theorem 1.

4.1 An Improved Derandomization Lemma for Interactive Proofs

In this section we prove a structure-preserving version of the well-known Babai-Moran "Speedup Theorem" [1,10] with improved parameters for our application. Essentially, the result says that for any constant-round public-coin interactive proof system with small soundness error, there exists a polynomial set of random

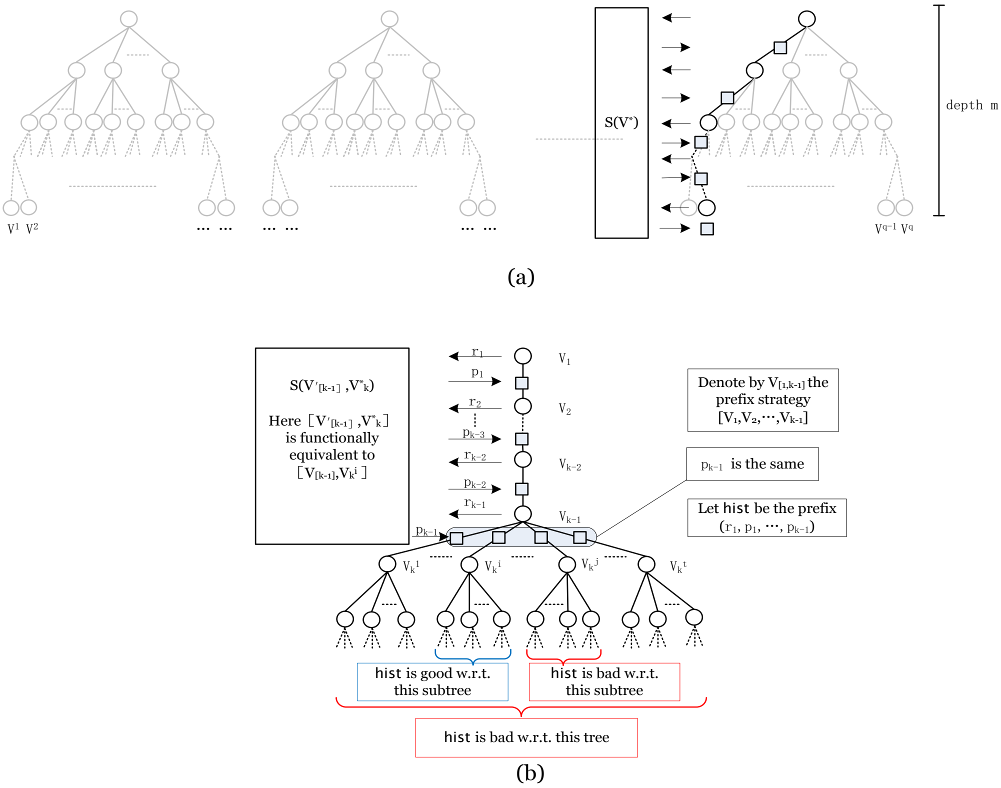

Fig. 3. Proof of Theorem 1. Figures (a) and (b) correspond to Lemma 2 and Lemma 3, respectively. In Figure (b), the prefix $(r_1, p_1, ..., p_{k-1})$ is bad w.r.t. the entire tree, which implies that there is a subtree (e.g., the one in the red underbrace) for which this session prefix is bad; however, the prefix is good w.r.t. the subtree that shares the same prefix strategy $[V_{[1,k-1]}, V_k^i]$ (the one in the blue underbrace) for which $V_k^* \overset{\mathrm{f}}{=} V_k^i$.

verifier tapes such that the cheating probability for the unbounded prover over these verifier tapes is bounded away from 1—and this holds even when the prover knows this small set of random tapes in advance.

We first recall the Babai-Moran theorem. Let $\mathrm{AM}[k]$ denote the set of languages whose membership can be proved via a k-round public-coin proof system.

Theorem 2 ([10]). *For any polynomial $t(n)$, $\mathrm{AM}[t+1] = \mathrm{AM}[t]$. In particular, for any constant k, $\mathrm{AM}[k] = \mathrm{AM}[2]$.*

For our application, we wish to de-randomize the verifier while keeping the original proof system structure intact (that is, without "collapsing" the round complexity). The $\mathrm{AM}[k] = \mathrm{AM}[2]$ proof—and its randomness-efficient variant in [11][4]—actually yield such a result: for any $2m$-round public-coin proof system with small soundness error ϵ, there exist $(O(p))^m$ verifier random tapes over which the cheating probability of an unbounded prover is still bounded away from 1, where p is the length of the prover's messages.

Next, we present an improvement to this result, in which the number of such verifier random tapes reduces to $(p/O(\log \frac{1}{\epsilon}))^m$. In addition, we show that this de-randomization lemma is essentially tight with respect to

[4] In [11], Bellare and Rompel present a randomness-efficient approach to transform $\mathrm{AM}[k]$ into $\mathrm{AM}[2]$: to halve the number of rounds of an Arthur-Merlin proof system, they introduce a so-called "oblivious sampler" and use a small amount of randomness to specify roughly $O(p)$ verifier messages in the original proof system. Their proof, however, yields almost the same result as the Speedup Theorem in our setting where we want to maintain the structure of the original proof system, and only care about the number of original verifier random tapes that are needed to make sure the resulting protocol after derandomization is still a proof system.

the round complexity, as there are super-constant-round public-coin proof systems for which the prover's cheating probability is 1, over any polynomial number of verifier random tapes.

Before stating the lemma, we introduce some additional notation:

- $V_{|(\mathbf{r}^1,\mathbf{r}^2...,\mathbf{r}^t)}$ denotes the honest verifier that is restricted to choose *uniformly at random* one of $\mathbf{r}^1, \mathbf{r}^2..., \mathbf{r}^t$ as its random tape, where t is a polynomial; we use $V_{|(\mathbf{r}^1,\mathbf{r}^2...,\mathbf{r}^t)}(\mathbf{r}^i)$ to denote the verifier that takes $\mathbf{r}^i$, $1 \le i \le t$, as its random tape.
- $P^*(\mathbf{r}^1, \mathbf{r}^2..., \mathbf{r}^t)$ denotes the unbounded *cheating* prover with auxiliary input $(\mathbf{r}^1, \mathbf{r}^2..., \mathbf{r}^t)$, indicating that it will interact with $V_{|(\mathbf{r}^1,\mathbf{r}^2...,\mathbf{r}^t)}$.

We now state the result formally. For simplicity, we assume that all the prover messages are of equal length.

Lemma 1. *Let m be a constant and $\langle P, V\rangle$ be a $2m$-round public-coin interactive proof system for language L with negligible soundness error ϵ. Let p denote the length of the prover's messages. Then for every $x \notin L$, there exist $q = (p/O(\log \frac{1}{\epsilon}))^m$ different random tapes, $\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q$, such that for every unbounded prover P,*

$$\Pr[\langle P(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q), V_{|(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)}\rangle(x) = 1] \le 1 - \frac{1}{q} .$$

Here we present the intuition and basic inequalities that yield the proof for the case of a 3-round proof system[5] (similar ideas also appeared in [1,10]), and defer the full proof of the lemma to Appendix A.1.

Let us consider a 3-round public-coin proof system $\langle P, V\rangle$ with negligible soundness error for some language L[6], in which the prover sends the first message p_1 and the last message p_2, and the verifier sends the second message $\mathbf{r}$ (its public coins). Without loss of generality, we assume $|p_1| = |p_2| = p$, and $|\mathbf{r}| = n$. We now prove that there exists a number p of verifier random tapes[7] $(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p)$ over which the cheating probability is at most $1 - 1/p$.

For the sake of contradiction, assume that for some false statement $x \notin L$ there is an unbounded prover $P^\diamond$ such that for any p-tuple $(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p)$, $P^\diamond(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p)$ can cheat $V_{|(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^p)}$ with probability 1. Now note that the number of such successful cheating provers is $\binom{2^n}{p}$, and that there are at most 2^p different first prover messages p_1. Thus, there is a number of at least $\binom{2^n}{p}/2^p$ $P^\diamond(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p)$'s that produce the same first prover message, denote it p_1^*, for which if the verifier is using a random tape in any of the p-tuples

$$\{(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p) : p_1^* \leftarrow P^\diamond(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p)\},$$

we have an unbounded prover that can produce a second prover message p_2^* to make the verifier accept.

On the other hand, the number of p-tuple choices $(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p)$ out of a $1/2e$ fraction of all possible verifier random tapes is at most $\binom{\frac{2^n}{2e}}{p}$. Since

$$\binom{\frac{2^n}{2e}}{p} < (\frac{2^n}{2p})^p < \frac{\binom{2^n}{p}}{2^p},$$

we have that the set $\{(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p) : p_1^* \leftarrow P^\diamond(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^p)\}$ covers at least a $1/2e$ fraction of all possible verifier random tapes.

In sum, we are able conclude that there is an unbounded prover, which sends p_1^* as its first message, that can make the verifier accept the false statement with probability at least $1/2e$. This contradicts the negligible soundness error of $\langle P, V\rangle$.

The proof of the lemma for the general (arbitrary constant rounds) case can be found in Appendix A.1, and the tightness result, i.e., the counterexample for superconstant-round proof systems, in Appendix B.

[5] The basic reasoning here applies to a proof system of even number (4) of rounds as well, by having the verifier send a dummy message first.

[6] For example, the n-folded parallel version of Blum's 3-round proof for Graph Hamiltonicity [8], or the 3-round proof for Graph Isomorphism [22].

[7] For simplicity's sake, we do not optimize this parameter here.

4.2 Proof of Theorem 1

Given Lemma 1, we now present the proof of Theorem 1. Again, let $\langle P, V\rangle$ be a $2m$-round public-coin canonical ZK proof for some non-trivial (outside $\mathcal{BPP}$) language L, and S be its associated simulator. We first prove the following claim, where Lemma 1 is used.

Lemma 2. *Let $\langle P, V\rangle$ be as above. Then there exist a false statement $x \notin L$ and q honest verifiers V^1, V^2, ..., V^q (recall that we use V^i as a shorthand for $V(\mathbf{r}^i)$, $1 \le i \le q$), such that given the description of any polynomial-time constructible $V^* \stackrel{\mathrm{f}}{=} V^i$ for a random i as input, the interaction $(S(V^*) \Leftrightarrow V^*)$[8] will produce an accepting transcript with probability negligibly close to 1, while the unbounded prover can cheat only with probability at most $1 - 1/q$.*

Proof. We first prove that there is some false statement $x \notin L$ so that for every PPT algorithm C which takes picks a random V from the set of verifiers and outputs V^* such that $V^* \stackrel{\mathrm{f}}{=} V$, the simulation $(S(V^*) \Leftrightarrow V^*)$ will generate an accepting transcript with probability negligibly close to 1 (over the randomness used by S and the random choice of verifier). Otherwise, if that were not the case, then the following simple algorithm could be used to decide membership in L efficiently[9]: Pick a verifier at random and run C to construct $V^* \stackrel{\mathrm{f}}{=} V$ as above, and then have S on input x and V^* interact with V^*; if V^* accepts, output "$x \in L$," otherwise output "$x \notin L$."

Now fix the above $x \notin L$, and set Q to be the set of verifier random tapes such that for any $\mathbf{r} \in Q$, and any polynomial-time constructible $V^* \stackrel{\mathrm{f}}{=} V(\mathbf{r})$, $(S(V^*) \Leftrightarrow V^*)$ will generate an accepting transcript with probability negligibly close to 1. We now show that the size of Q is larger than a $(1 - \mathsf{neg}(n))$ fraction of all possible random tapes. Assume the verifier's random tape $\mathbf{r}$ and S's random tape R are uniformly distributed over $\{0,1\}^l$ and $\{0,1\}^s$, respectively, where l and s are some polynomials, and denote by E the event that the simulation $(S(V^*) \Leftrightarrow V^*)$ generates an accepting transcript. We have

$$\begin{aligned}
& \Pr_{\substack{\mathbf{r}\leftarrow\{0,1\}^l \\ R\leftarrow\{0,1\}^r}} [V^* \leftarrow C(\mathbf{r}) : E] \qquad (1)\\
= & \Pr_{\substack{\mathbf{r}\leftarrow\{0,1\}^l \\ R\leftarrow\{0,1\}^r}} [V^* \leftarrow C(\mathbf{r}) : E | \mathbf{r} \in Q] \Pr[\mathbf{r} \in Q] \\
& + \Pr_{\substack{\mathbf{r}\leftarrow\{0,1\}^l \\ R\leftarrow\{0,1\}^r}} [V^* \leftarrow C(\mathbf{r}) : E | \mathbf{r} \notin Q] \Pr[\mathbf{r} \notin Q] \\
\le & \Pr[\mathbf{r} \in Q] + (1 - \frac{1}{\mathsf{poly}(n)}) \Pr[\mathbf{r} \notin Q] \\
= & \frac{|Q|}{2^l} + (1 - \frac{1}{\mathsf{poly}(n)})(1 - \frac{|Q|}{2^l}) \\
= & 1 - \frac{1}{\mathsf{poly}(n)}(1 - \frac{|Q|}{2^l}).
\end{aligned}$$

Given that the probability in expression (1) is greater than $1 - \mathsf{neg}(n)$, so is the quantity $\frac{|Q|}{2^l}$.

Thus, given $x \notin L$, for any unbounded prover, the cheating probability, taken over the choices of verifier random tapes in Q, is still negligible. Applying now Lemma 1, we can find q random tapes $\mathbf{r}_i \in Q$, $1 \le i \le q$, such that the probability, taken over these q random tapes, that the unbounded prover makes the verifier accept is at most $1 - 1/q$. This completes the proof of the lemma. □

The next lemma, where Lemma 2 is used, is the key step in establishing our main theorem.

[8] Recall that this interaction is identical to $(S(V^*) \Leftrightarrow V^i)$.

[9] Although the error probability here may be high, it can be reduced by standard parallel repetition.

Lemma 3. *Let $\langle P, V\rangle$ be as above and fix the false statement x. Then there exists a triplet $(k, \mathcal{V}_{[1,k-1]}, V'_{[1,k-1]})$, where:*

- $2 \leq k \leq m$;
- $\mathcal{V}_{[1,k-1]}$ *is a subset of verifiers that share the same prefix strategy $V_{[1,k-1]}$ but have t distinct k-th step strategies $V_k^1, V_k^2, ..., V_k^t$, denoted by $\mathcal{V}_k$ (we let $\mathcal{V}^i_{[1,k]}$ denote the subset of verifiers in $\mathcal{V}_{[1,k-1]}$ that share the same prefix strategy $[V_{[1,k-1]}, V_k^i]$); and*
- $V'_{[1,k-1]}$ *is a prefix verifier code functionally equivalent to $V_{[1,k-1]}$,*

such that, for any $1 \leq i \leq t$ and any polynomial-time constructible code V_k^* satisfying $V_k^* \stackrel{\mathrm{f}}{=} V_k^i$, $(S([V'_{[1,k-1]}, V_k^*]) \Leftrightarrow [V'_{[1,k-1]}, V_k^*])$ will generate a session prefix $(r_1, p_1, ..., p_{k-1})$ satisfying the following two conditions:

1. $(r_1, p_1, ..., p_{k-1})$ *is* bad *with respect to* $\mathcal{V}_{[1,k-1]}$;
2. $(r_1, p_1, ..., p_{k-1})$ *is* good *with respect to* $\mathcal{V}^i_{[1,k]}$.

We prove the lemma by examining the next-message functions of the q honest verifiers $V^1, V^2, ...V^q$ guaranteed by Lemma 2, step by step. At a high level, the structure of the proof is as follows:

1. First, show that there exists a triplet $(2, \mathcal{V}_{[1]}, V'_{[1]})$ satisfying condition 1.
2. Show that any $(m-1, \mathcal{V}_{[1,m-1]}, V'_{[1,m-1]})$ satisfies condition 2.
3. Show that, for any $2 \leq k \leq m-1$, if a given $(k, \mathcal{V}_{[1,k-1]}, V'_{[1,k-1]})$ satisfes condition 1, but not condition 2, then we have a triplet $(k+1, \mathcal{V}_{[1,k]}, V'_{[1,k]})$ that satisfies condition 1.

This reasoning guarantees that we can find a triplet $(k, \mathcal{V}_{[1,k-1]}, V'_{[1,k-1]})$, for some $2 \leq k \leq m$, which satisfies both conditions. The detailed proof of the above three steps is presented in Appendix A.2.

We are now ready to construct the understanding oracle algorithm U^S, yielding the proof of the theorem. Fix the false statement x, k, $\mathcal{V}_{[1,k-1]}$, $\mathcal{V}_k$, and $V'_{[1,k-1]}$ as in Lemma 3. Let the output of an arbitrary PPT algorithm C on input $(\mathcal{V}_k, i)$ for random i be V_k^* such that $V_k^* \stackrel{\mathrm{f}}{=} V_k^i \in \mathcal{V}_k$. Algorithm U^S works as follows.[10]

The *understanding* algorithm U^S.

Input to U: $\mathcal{V}_{[1,k-1]}$, $\mathcal{V}_k$ and an initially empty set T.
Oracle access to $S([V'_{[1,k-1]}, V_k^*])$.

1. Play the role of a verifier using the prefix strategy $V_{[1,k-1]}$ to interact with $S([V'_{[1,k-1]}, V_k^*])$[11] until obtaining a session prefix $(r_1, p_1, ..., p_{k-1})$.
2. For each j, $1 \leq j \leq t$, exhaust all possible prover messages after the k-th verifier step, checking if the session prefix $(r_1, p_1, ..., p_{k-1})$ is good with respect to $\mathcal{V}_k^j$. If not, add j to set T.
3. Output an arbitrary j in T.

As mentioned before, in its second step, U can check whether the given session prefix is good in time $2^{O(p)}$, which overwhelmingly dominates its the running time.

Condition 1 of Lemma 3 guarantees that there exists j such that the session prefix $(r_1, p_1, ..., p_{k-1})$ produced in U's step 1 is *bad* with respect to $\mathcal{V}^j_{[1,k-1]}$, which implies that T is not empty. Condition 2 of Lemma 3 guarantees that if $(r_1, p_1, ..., p_{k-1})$ is bad with respect to $\mathcal{V}^j_{[1,k-1]}$, then $V_k^* \stackrel{\mathrm{f}}{\neq} V_k^j$. In other words, algorithm U was able to pin-point a program (V_k^j) functionally *different* from V_k^*. This concludes the proof of the theorem.

[10] Keep in mind that we omit inputs x and randomness to S and U for simplicity.
[11] Again, by Fact 1 (Section 2), this is equivalent to the interaction between $V'_{[1,k-1]}$ and $S([V'_{[1,k-1]}, V_k^*])$.

5 Conclusions

A natural question which arises from our reduction is: How hard is the verifier-understanding problem? Note that what we wish to understand is the partial code of an honest verifier algorithm, which is simply a set of *constant functions*, since the proof system is public-coin. Thus, if S actually runs this code internally, it can figure out the functionality of the target code of the partial verifier fairly easily, and thus our understanding problem should be solvable in moderately exponential time by U^S.

Observe though that the understanding algorithm only obtains from $S(V'_{[1,k-1]}, V_k^*)$ prefix $r_1, p_1, ..., p_{k-1}$, and not r_k, which is supposed to be V_k^*'s output. Thus, if during the interaction between U and S, S already ran V^* internally, then this means that S rewinds V_k^* since it runs V_k^* *before* generating the session prefix up to the prover's $(k-1)$-th step.

It is hard to imagine that one could adopt a rewinding strategy in order to simulate a malicious verifier for a constant-round public-coin ZK proof system. This actually leads us to think of our main theorem as evidence against the existence of such proof systems: if S, in the interaction with U, does *not* run the target code V_k^* at all (neither does U^S—recall that U is not even being given this target code), then it seems very unlikely for such U^S to be able to solve the verifier-understanding problem with respect to an *arbitrary* polynomial-time-constructible code, even in time exponential in the length of prover message[12]. As such, our result constitutes very strong evidence against the existence of such proof systems admitting so-called straight-line simulators, in sharp contrast to Barak's construction of a constant-round public-coin ZK argument, whose simulator indeed runs in a straight-line fashion.

References

[1] L. Babai: Trading Group Theory for Randomness. STOC 1985, pp. 421-429, 1985.

[2] B. Barak: How to go beyond the black-box simulation barrier. FOCS 2001, pp.106-115.

[3] G. Brassard, D. Chaum, and C. Crépeau: Minimum disclosure proofs of knowledge. J. Comput. Syst. Sci., 37(2):156-189, 1988.

[4] N. Bitansky, D. Dachman-Soled, S. Garg, A. Jain, Y. Kalai, A. Lpez-Alt, D. Wichs: Why "Fiat-Shamir for Proofs" Lacks a Proof. TCC 2013: 182-201.

[5] B. Barak, O. Goldreich, S. Goldwasser, Y. Lindell: Resettably sound ZK and its Applications. FOCS 2001, pp. 116-125, 2001.

[6] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, K. Yang: On the (Im)possibility of Obfuscating Programs. CRYPTO 2001, pp.1-18, 2001.

[7] B. Barak, Y. Lindell: Strict polynomial-time in simulation and extraction. STOC 2002, pp.484-493. 2002.

[8] M. Blum: How to prove a theorem so no one else can claim it. Proceedings of theInternational Congress of Mathematicians, pp.444-451, 1986.

[9] B. Barak, Y. Lindell, S. P. Vadhan: Lower Bounds for Non-Black-Box Zero Knowledge. FOCS 2003, pp.384-393,2003.

[10] L. Babai, S. Moran: Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. J. Comput. Syst. Sci. 36(2): 254-276, 1988.

[11] M. Bellare, J. Rompel: Randomness-Efficient Oblivious Sampling FOCS 1994: 276-287.

[12] R. Canetti and R. R. Dakdouk: Obfuscating Point Functions with Multibit Output. EUROCRYPT 2008, pp.489-508, 2008.

[13] R. Canetti, O. Goldreich, S. Goldwasser, S. Micali. Resettable Zero Knowledge. STOC 2000, pp.235-244, 2000.

[14] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Concurrent Zero-Knowledge requires $\Omega(logn)$ rounds. STOC 2001, pp.570-579, 2001.

[15] I. Damgard. Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. EUROCYPT 2000, pp.174-187, 2000.

[16] Y. Deng, V. Goyal, A. Sahai: Resolving the Simultaneous Resettability Conjecture and a New Non-Black-Box Simulation Strategy. FOCS 2009: 251-260

[17] G. Di Crescenzo, Ivan Visconti. Concurrent ZK in the Public-Key Model. ICALP 2005, pp.816-827, 2005.

[18] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. STOC 1998, pp.409-418, 1998.

[19] U. Feige, D. Lapidot and A. Shamir: Multiple Non-Interactive Zero Knowledge Proofs Under General Assumptions. SIAM J. on Computing 29 (1999) 1C28.

[20] O. Goldreich: The Foundations of Cryptography - Volume 1, Basic Techniques. Cambridge University Press 2001.

[21] O. Goldreich and H. Krawczyk: On the Composition of Zero-Knowledge Proof Systems. SIAM J. Comput. 25(1), pp.169-192, 1996.

[22] O. Goldreich, S. Micali and A. Wigderson. Proofs that yield nothing but their validity or All languages in NP have zero-knowledge proof systems. J. ACM, 38(3), pp.691-729, 1991.

[12] We note that U's running time is *independent* of the length of the target code, making the understanding task even harder.

[23] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM. J. Computing, 18(1):186-208, February 1989.
[24] O. Goldreich, S. Vadhan and A. Wigderson. On Interactive Proofs with a Laconic Prover. ICALP 2001, pp. 334-345.
[25] S. Goldwasser, Guy N. Rothblum: On Best-Possible Obfuscation. TCC 2007, pp.194-213, 2007.
[26] S. Hada: Zero-Knowledge and Code Obfuscation. ASIACRYPT 2000, pp.443-457, 2000.
[27] S. Hohenberger, Guy N. Rothblum, A. Shelat, V. Vaikuntanathan: Securely Obfuscating Re-encryption. TCC 2007, pp.233-252, 2007.
[28] R. Pass, A. Rosen: New and improved constructions of non-malleable cryptographic protocols. STOC 2005: 533-542.
[29] M. Prabhakaran, A. Rosen, A. Sahai: Concurrent Zero Knowledge with Logarithmic Round-Complexity. FOCS 2002, pp.366-375, 2002.
[30] H. Wee: On obfuscating point functions. STOC 2005, pp.523-532, 2005.

A Proofs

A.1 Proof of Lemma 1

We first introduce some definitions and additional notation that will be used in the proof.

We assume that the length of each prover message is greater than any constant, in particular, $p > 10$. Note that this assumption is without loss of generality because if the length of the prover message in a constant-round interactive proof for a language L is constant, then L is trivial (see [24]), which in turn implies our lemma immediately.

Throughout this subsection, we consider only *structured* q-tuples of verifier's random tapes, which are selected in the following way:

1. For each verifier step i, $1 \le i \le m$, if $|\{0,1\}^{l_i}| > \frac{m^2 p}{2\log\frac{1}{\epsilon}}$, set $t_i = \frac{m^2 p}{2\log\frac{1}{\epsilon}} \in p/O(\log\frac{1}{\epsilon})$; otherwise, set $t_i = 2^{l_i}$, where l_i is the length of the i-th verifier message;
2. Choose t_i *distinct* strings $r_{1i}, r_{2i}, ..., r_{t_i i}$ from $\{0,1\}^{l_i}$;
3. Choose an i-th verifier message $r_{j_i i} \in (r_{1_i i}, r_{2i}, ..., r_{t_i i})$, $1 \le j_i \le t_i$ for each step i, and set random tape $\mathbf{r}^j = [r_{j_1 1}, r_{j_2 2}, ..., r_{j_m m}]$.
4. A q-tuple of random tapes is now the set of all possible random tapes set in step 3, $(\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^q)$. Note that the size q of this set is $\prod_{i=1}^m t_i$, which is determined by Step 2.

We identify $(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ with $(\mathbf{r}^{\pi(1)}, \mathbf{r}^{\pi(2)}, ...\mathbf{r}^{\pi(q)})$ for any permutation π on $\{1, 2, ..., q\}$. Two q-tuples, $(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ and $(\mathbf{r'}^1, \mathbf{r'}^2, ...\mathbf{r'}^q)$, are said to be *distinct* if there exists at least one $\mathbf{r}^i$ such that $\mathbf{r}^i \in (\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ but $\mathbf{r}^i \notin (\mathbf{r'}^1, \mathbf{r'}^2, ...\mathbf{r'}^q)$, or vice-versa. Thus the number of all possible distinct such structured q-tuples is

$$\prod_{i=1}^{m} \binom{2^{l_i}}{t_i}.$$

Some more basic notation before the proof:

- $\text{prefix}_i(\mathbf{r}^j)$: the first i messages from the verifier using random tape $\mathbf{r}^j$, that is, for $\mathbf{r}^j = [r_1^j, r_2^j, ..., r_m^j]$, $\text{prefix}_i(\mathbf{r}^j) = [r_1^j, r_2^j, ..., r_i^j]$.
- $\overrightarrow{T}$ and its size $|\overrightarrow{T}|$: $\overrightarrow{T}$ is a *set* of structured q-tuples of verifier's random tapes, The size of $\overrightarrow{T}$, denoted by $|\overrightarrow{T}|$, is simply defined to be the number of *distinct* q-tuples in $\overrightarrow{T}$.
- $p_k \leftarrow \langle P(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q), V\rangle_{|\mathsf{hist}}$ denotes the k-th prover message produced by the prover $P(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ (the prover strategy taking q-tuple $(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ as auxiliary input), conditioned on hist being the current history so far.

The proof of the lemma is by contradiction. Assume that there exists an unbounded prover, call it $P^\diamond$, and $x \notin L$, such that for any q-tuple $(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$, $r_i \ne r_j$ for $i \ne j$:

$$\Pr[\langle P^\diamond(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q), V_{|(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)}\rangle(x) = 1] > 1 - \frac{1}{q}\ . \tag{2}$$

First note that $V_{|(\mathbf{r}^1,\mathbf{r}^2,...,\mathbf{r}^t)}(\mathbf{r}^i)$ acts exactly the same as $V(\mathbf{r}^i)$. Therefore

$$\Pr[\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q), V_{|(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)}\rangle(x)=1] \tag{3}$$

$$= \sum_i \Pr[\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q), V_{|(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)}(\mathbf{r}^i)\rangle(x)=1]\frac{1}{q} \tag{4}$$

$$= \sum_i \Pr[\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q), V(\mathbf{r}^i)\rangle(x)=1]\frac{1}{q}\ . \tag{5}$$

Further, observe that the probability $\Pr[\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q), V(\mathbf{r}^i)\rangle(x)=1]$ is either 0 or 1 because in this interaction the tapes are fixed and both prover and verifier are deterministic. Thus, if inequality (2) holds, we have

$$\Pr[\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q), V_{|(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)}\rangle(x)=1]=1\ , \tag{6}$$

and, by (5),

$$\Pr[\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q), V(\mathbf{r}^i)\rangle(x)=1]=1\ . \tag{7}$$

Now, given prover $P^\diamond$ such that (7) holds for any q-tuple $(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)$, we describe a prover P^* that will cheat V with probability greater than ϵ.

The Cheating Prover P^*.

Input: x, as in inequality (2).

1. Set $\overrightarrow{T^0}$ to be the set of all possible distinct structured q-tuples over $\{0,1\}^{l_1+l_2+...+l_m}$, and G_1 the set of all possible first verifier's messages (i.e., the set $\{0,1\}^{l_1}$).
2. For $k=1$ to m, do

   2.1. Upon receiving the k-th verifier message r_k, set hist to be the current history $(r_1,p_1^*,...,r_k)$. Check if $r_k\in G_k$. If $r_k\notin G_k$, abort and output "$\perp$". Otherwise, for every q-tuple $(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)\in\overrightarrow{T^0}$ such that: a) it contains some $\mathbf{r}^i$ such that $\text{prefix}_{k-1}(\mathbf{r}^i)=[r_1,r_2,...,r_k]$, and, b) the current hist is consistent with the interaction between $P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)$ and V, set $t'=\prod_{k+1}^m t_i$, compute the k-th prover message by running $P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)$, and obtain the set of k-th prover messages

$$\{p_k\leftarrow\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q),V\rangle_{|\mathsf{hist}}:$$
$$(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)\in\overrightarrow{T^0}\text{ and }\exists(\mathbf{r}^{i_1},\mathbf{r}^{i_2},...\mathbf{r}^{i_{t'}})\in(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)\text{ s.t.}$$
$$\text{prefix}_k(\mathbf{r}^{i_j})=[r_1,r_2,...,r_k]\text{ for all }1\le j\le t'=\prod_{k+1}^m t_i\}^{13}\ .$$

   Set p_k^* to be the p_k that maximizes the size of the set

$$\{(\mathbf{r}^{i_1},\mathbf{r}^{i_2},...\mathbf{r}^{i_{t'}}):p_k\leftarrow\langle P^\diamond(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q),V\rangle_{|\mathsf{hist}},\text{ and}$$
$$(\mathbf{r}^{i_1},\mathbf{r}^{i_2},...\mathbf{r}^{i_{t'}})\in(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q),\text{ and}$$
$$\text{prefix}_k(\mathbf{r}^{i_j})=[r_1,r_2,...,r_k]\text{ for all }1\le j\le t'=\prod_{k+1}^m t_i\}$$

[13] Observe that, by the structure of q-tuple, if there exists a $\mathbf{r}^i\in(\mathbf{r}^1,\mathbf{r}^2,...\mathbf{r}^q)$ such that $\text{prefix}_k(\mathbf{r}^i)=[r_1,r_2,...,r_k]$, then there exist $t'=\prod_{k+1}^m t_i$ many such random tapes.

2.2. If $k < m$, denote by $\overrightarrow{T^k}$ the above set that achieves its maximum size, and set (guessing the next verifier messages)

$$G_{k+1} \leftarrow \{r_{k+1} \in \{0,1\}^{l_{k+1}} : |\{(\mathbf{r}^{i_1}, \mathbf{r}^{i_2}, ...\mathbf{r}^{i_{t'}}) \in \overrightarrow{T^k} :$$

$$\text{prefix}_{k+1}(\mathbf{r}^{i_j}) = [r_1, r_2, ..., r_k, r_{k+1}] \text{ for all } 1 \leq j \leq t' = \prod_{k+1}^{m} t_i\}| \geq \frac{\prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp}}\}.$$

In a nutshell, the above algorithm just tries many different cheating provers $P^{\diamond}(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ to make the current history accepted by as many verifiers as possible.

Analysis of algorithm P^*. Let us now analyze the success probability of the prover's strategy outlined above. We first show that the size of G_k is large enough for every k.

Claim. For every $1 \leq k \leq m$, conditioned on P^* not outputting $\perp$, $|G_k| \geq \frac{2^{l_k}}{2^{1.1kp/t_k}e}$.

Proof. When $k = 1$, $|G_1| = |\{0,1\}^{l_1}| > \frac{2^{l_k}}{2^{1.1kp/t_k}e}$.

When $k \geq 2$, the condition of P^* not outputting "$\perp$" implies that, for $j \leq k$, r_j is in G_j, and that

$$|\{(\mathbf{r}^{i_1}, \mathbf{r}^{i_2}, ...\mathbf{r}^{i_{t''}}) \in \overrightarrow{T^{k-1}} : \text{prefix}_k(\mathbf{r}^{i_j}) = [r_1, r_2, ..., r_k] \text{ for all } 1 \leq j \leq t'' = \prod_{k}^{m} t_i\}| \geq$$

$$\frac{\prod_{i=k}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1(k-1)p}}.$$

which in turn leads to (recall that the length of prover messages is p), for $k \geq 2$,

$$|\overrightarrow{T^k}| \geq \frac{\prod_{i=k}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1(k-1)p+p}} = \frac{\prod_{i=k}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp-0.1p}}. \tag{8}$$

Now assume that, for $k \geq 2$, conditioned on P^* not outputting "$\perp$" (i.e., for $j \leq k$, r_j is in G_j), $|G_k| < \frac{2^{l_k}}{2^{1.1kp/t_k}e}$.

Set $t' = \prod_{k+1}^{m} t_i$. Recall that all t'-tuples $(\mathbf{r}^{i_1}, \mathbf{r}^{i_2}, ..., \mathbf{r}^{i_{t'}}) \in \overrightarrow{T^k}$ share the same prefix $[r_1, r_2, ..., r_k]$, and that, by the structure of q-tuple of random tapes, within a t'-tuple $(\mathbf{r}^{i_1}, \mathbf{r}^{i_2}, ..., \mathbf{r}^{i_{t'}}) \in \overrightarrow{T^k}$, there are only t_k distinct k-th verifier messages, say $(r_k^1, r_k^2, ..., r_k^{t_k})$. We partition these t'-tuples in $\overrightarrow{T^k}$ in two classes by the property of $(r_k^1, r_k^2, ..., r_k^{t_k})$:

1. Every $r_k^i \in (r_k^1, r_k^2, ..., r_k^{t_k})$ is in G_k (which implies $t_k \leq |G_k|$). The number of t'-tuples in $\overrightarrow{T^k}$ satisfying this condition is at most
$$\binom{|G_k|}{t_k} \prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i}.$$

2. There is at least one $r_k^i \in (r_k^1, r_k^2, ..., r_k^{t_k})$ that is *not* in G_k. Then by the definition of G_k, and by the fact that, within a t'-tuple $(\mathbf{r}^{i_1}, \mathbf{r}^{i_2}, ..., \mathbf{r}^{i_{t'}}) \in \overrightarrow{T^k}$, for every i, the number of random tapes in this t'-tuple with each prefix $[r_1, r_2, ..., r_{k-1}, r_k^i]$ is the same (equal to $\prod_{k+1}^{m} t_i$), then the number of t'-tuples in $\overrightarrow{T^k}$ satisfying this condition is at most
$$\binom{2^{l_k}}{t_k} \frac{\prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp}}.$$

Thus, we have

$$
\begin{aligned}
|\overrightarrow{T^k}| &\leq \binom{|G_k|}{t_k} \prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i} + \binom{2^{l_k}}{t_k} \frac{\prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp}} \\
&< \binom{\frac{2^{l_k}}{2^{1.1kp/t_k} e}}{t_k} \prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i} + \binom{2^{l_k}}{t_k} \frac{\prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp}} \\
&< (\frac{2^{l_k}}{2^{1.1kp/t_k} t_k})^{t_k} \prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i} + \binom{2^{l_k}}{t_k} \frac{\prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp}} \\
&< \frac{(\frac{2^{l_k}}{t_k})^{t_k}}{2^{1.1kp}} \prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i} + \binom{2^{l_k}}{t_k} \frac{\prod_{i=k+1}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp}} \\
&< \frac{\prod_{i=k}^{m} \binom{2^{l_i}}{t_i}}{2^{1.1kp-1}},
\end{aligned}
$$

which contradicts (8) when $p > 10$, which we can always assume without loss of generality (otherwise our lemma holds trivially; see [24]). □

Now observe that, for every prover step $k \leq m$, if $G_k \geq \frac{2^{l_k}}{2^{1.1kp/t_k} e}$, then the probability that P^* guesses the next verifier message correctly, i.e., the probability that $r_k \in G_k$, is $|G_k|/2^{l_k} = \frac{1}{2^{1.1kp/t_k} e}$. Therefore P^* guesses all the next verifier messages correctly with probability at least

$$\prod_{k=1}^{m} \frac{|G_k|}{2^{l_k}} = \prod_{k=1}^{m} \frac{1}{2^{1.1kp/t_k} e},$$

which is greater than ϵ for $t_k \leq \frac{m^2 p}{2 \log \frac{1}{\epsilon}}$. (Recall that either $t_k = \frac{m^2 p}{2 \log \frac{1}{\epsilon}}$, or $t_k = 2^{l_k}$ when $2^{l_k} \leq \frac{m^2 p}{2 \log \frac{1}{\epsilon}}$.)

Notice also that, in case that all guesses of the next verifier messages are correct, there exists at least one q-tuple $(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ such that the complete transcript $(r_1, p_1^*...r_m, p_m^*)$ is generated in the interaction between $P^\diamond(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ and $V(\mathbf{r}^i)$, $\mathbf{r}^i = [r_1, r_2...r_m] \in (\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$, which is guaranteed by our assumption to be accepting.

In sum, our cheating prover P^* will cheat with probability greater than ϵ, which breaks the soundness of the proof system $\langle P, V \rangle$, thus yielding the lemma.

A.2 Proof of Lemma 3

We prove the lemma by examining the next-message functions of the q honest verifiers $V^1, V^2, ...V^q$ guaranteed by Lemma 2, step by step. Recall that the structure of the proof is as follows:

1. First, show that there exists a triplet $(2, \mathcal{V}_{[1]}, V'_{[1]})$ satisfying condition 1.
2. Show that any $(m-1, \mathcal{V}_{[1,m-1]}, V'_{[1,m-1]})$ satisfies condition 2.
3. Show that, for any $2 \leq k \leq m-1$, if a given $(k, \mathcal{V}_{[1,k-1]}, V'_{[1,k-1]})$ satisfes condition 1, but not condition 2, then we have a triplet $(k+1, \mathcal{V}_{[1,k]}, V'_{[1,k]})$ that satisfies condition 1.

This reasoning guarantees that we can find a triplet $(k, \mathcal{V}_{[1,k-1]}, V'_{[1,k-1]})$, for some $2 \leq k \leq m$, which satisfies both conditions. We now turn to proving the above three steps.

The proof of step 1 is as follows. By Lemma 2, no unbounded prover can cheat a random verifier from set $\{V^1, V^2, ...V^q\}$ with probability 1. This immediately means (recall that we assume that verifier sends the first message in a session) that there exists V_1 such that no unbounded prover can cheat a random verifier having the same prefix strategy V_1 chosen from $\{V^1, V^2, ...V^q\}$ with probability 1.

Thus, we can have $(2, \mathcal{V}_{[1]}, V'_{[1]})$, where $\mathcal{V}_{[1]}$ is the set of verifiers in $\{V^1, V^2, ...V^q\}$ having the same prefix strategy V_1 and the code V'_1 is V_1. By the structure of these q verifiers, we have that $\mathcal{V}_{[1]}$ has a set $\mathcal{V}_2$ of t distinct second-step strategies $V_2^1, V_2^2, ..., V_2^t$. It is easy to see that the first condition of the lemma now holds, as otherwise, if there exists an i and code $V_k^* \stackrel{\text{f}}{=} V_2^i$, such that the session prefix $(r_1, p_1) \leftarrow (S([V'_1, V_2^*]) \Leftrightarrow [V'_1, V_2^*])$ is good with respect to $\mathcal{V}_{[1]}$, then the following unbounded prover with auxiliary input $[V'_1, V_2^*]$ and $\mathcal{V}_{[1]}$ will cheat a random verifier V in $\mathcal{V}_{[1]}$ with probability 1: Upon receiving the first verifier message (produced by V_1), it runs $S([V'_1, V_2^*])$, obtains p_1, and then runs the residual prover strategy guaranteed to exist by the definition of a good session prefix (Definition 5) to complete the interaction with V.

Step 2 is guaranteed by Lemma 2. Given any $(m-1, \mathcal{V}_{[1,m-1]}, V'_{[1,m-1]})$, where $\mathcal{V}_{[1,m-1]}$ shares the same prefix strategy $V_{[1,m-1]}$ but has t distinct m-th step strategies $V_m^1, V_m^2, ..., V_m^t$, and $V'_{[1,m-1]} \stackrel{\text{f}}{=} V_{[1,m-1]}$, the reason for this triplet satisfying condition 2 is that, for any i and any polynomial-time constructible code V_m^* satisfying $V_m^* \stackrel{\text{f}}{=} V_m^i$, the session prefix $(r_1, p_1, ..., p_m) \leftarrow (S([V'_{[1,m-1]}, V_m^*]) \Leftrightarrow [V'_{[1,m-1]}, V_m^*])$ must be good since $(r_1, p_1, ..., p_m)$ is, by the property of the simulator guaranteed by Lemma 2, an accepting and complete transcript.

We now prove step 3 using the canonical property of ZK proofs (Definition 6). Assume there is a triplet $(k, \mathcal{V}_{[1,k-1]}, V'_{[1,k-1]})$, $2 \leq k \leq m-1$ (where again $\mathcal{V}_{[1,k-1]}$ shares the same prefix strategy $V_{[1,k-1]}$ but has t distinct k-th step strategies $V_k^1, V_k^2, ..., V_k^t$, and $V'_{[1,k-1]} \stackrel{\text{f}}{=} V_{[1,k-1]}$), which satisfies condition 1, but not condition 2. Note that conditioned on not satisfying condition 2, we have an i such that for any code $V_k^* \stackrel{\text{f}}{=} V_k^i$, the session prefix $(r_1, p_1, ..., p_k) \leftarrow (S([V'_{[1,k-1]}, V_k^*]) \Leftrightarrow [V'_{[1,k-1]}, V_k^*])$ is *bad* with respect to the set of verifiers in $\mathcal{V}_{[1,k]}$ having the same prefix strategy $[V_{[1,k-1]}, V_k^i]$ (again, $\mathcal{V}_{[1,k]}$ has t distinct $(k+1)$-th step strategies $V_{k+1}^1, V_{k+1}^2, ..., V_{k+1}^t$).

By setting $V'_{[1,k]}$ to be $[V'_{[1,k-1]} and V_k^*]$, $\mathcal{V}_{[1,k]}$ as above, we now have a triplet $(k+1, \mathcal{V}_{[1,k]}, V'_{[1,k]})$ for which the condition 1 holds, for the following reason: Assume otherwise, i.e., that there exist V_{k+1}^i, and a code $V_{k+1}^* \stackrel{\text{f}}{=} V_{k+1}^i$ such that $(r_1, p'_1, ..., p'_k) \leftarrow (S([V'_{[1,k]}, V_{k+1}^*]) \Leftrightarrow [V'_{[1,k]}, V_{k+1}^*])$ is good with respect to $\mathcal{V}_{[1,k]}$. Then, by the canonical property (Definition 6), $(r_1, p_1, ..., p_k) \leftarrow (S(V'_{[1,k]}) \Leftrightarrow V'_{[1,k]})$ is also good with respect to $\mathcal{V}_{[1,k]}$, which contradicts the assumption that $(k, \mathcal{V}_{[1,k-1]}, V'_{[1,k-1]})$ does not satisfy condition 2.

B Interactive Proof Systems with Super-Constant Rounds

In this section we give a simple super-constant-round public-coin interactive proof system for which Lemma 1 does not hold.

Preamble: For $1 \leq k \leq s$, do:
$P \to V$: Send n random strings p_1^k,...,p_n^k of length n each.
$V \to P$: Send a random string r_k of length n.
Main proof: If there is some $p_i^k = r_k$, V accepts; otherwise execute a 3-round Blum protocol [8] with negligible soundness error.

Observe that for any q, if q different verifier random tapes $(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ are fixed in advance and known to an all-powerful prover, then for the cheating probability to be strictly less than 1, at any verifier step $k \leq s$, given $(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q)$ and current history hist), there must be at least $n+1$ possible different verifier next messages (i.e., the entropy $H(r_k|(\mathbf{r}^1, \mathbf{r}^2, ...\mathbf{r}^q), \mathsf{hist})$ is greater than $\log n$), which leads to $q \geq (n+1)^s$. That is, if s is super-constant, for any polynomial number of verifier's random tapes that are fixed in advance we have a prover with cheating probability 1.