DEFINITIONS AND PROPERTIES OF ZERO-KNOWLEDGE PROOF SYSTEMS

Oded Goldreich
Yair Oren

Department Of Computer Science
Technion, Haifa, Israel

Abstract

In this paper we investigate some properties of zero-knowledge proofs, a notion introduced by Goldwasser, Micali and Rackoff. We introduce and classify two definitions of zero-knowledge: $auxiliary - input$ zero-knowledge and $blackbox - simulation$ zero-knowledge. We explain why auxiliary-input zero-knowledge is a definition more suitable for cryptographic applications than the original [GMR1] definition. In particular, we show that any protocol solely composed of subprotocols which are auxiliary-input zero-knowledge is itself auxiliary-input zero-knowledge. We show that blackbox-simulation zero-knowledge implies auxiliary-input zero-knowledge (which in turn implies the [GMR1] definition). We argue that all known zero-knowledge proofs are in fact blackbox-simulation zero-knowledge (i.e., were proved zero-knowledge using blackbox-simulation of the verifier). As a result, all known zero-knowledge proof systems are shown to be auxiliary-input zero-knowledge and can be used for cryptographic applications such as those in [GMW2]. We demonstrate the triviality of certain classes of zero-knowledge proof systems, in the sense that only languages in BPP have zero-knowledge proofs of these classes. In particular, we show that any language having a *Las Vegas* zero-knowledge proof system necessarily belongs to RP. We show that randomness of both the verifier and the prover, and non-triviality of the interaction are essential properties of (non-trivial) auxiliary-input zero-knowledge proofs.

Preliminary versions of this work have appeared in [O1, O2].

WARNING: The current text was automatiocally translated from old troff files. Such translations may introduce errors. Furthermore, I'm not sure whether the source troff files I've found are actually the onesw corresponding to the final version. Errors may be due to this fact too. The final version has appeared in *Journal of Cryptology*, Vol. 7, No. 1 (1994), pp. 1–32.

1. INTRODUCTION The fundamental notion of zero-knowledge was introduced by Goldwasser, Micali and Rackoff in [GMR1]. They considered a setting where a powerful *prover* is proving a theorem to a probabilistic polynomial time *verifier*. Intuitively, a proof system is considered zero-knowledge if whatever the verifier can compute while interacting with the prover it can compute by itself without going through the protocol. The intriguing nature of this notion has raised considerable interest and many questions to be answered. Zero-knowledge proofs are of wide applicability in the field of cryptographic protocols, as demonstrated by Goldreich, Micali and Wigderson in [GMW1, GMW2]. In this paper we investigate some aspects of these proof systems. We present new definitions of zero knowledge, discuss their importance, and investigate their relative power. In the second part of the paper we demonstrate that certain properties are essential to zero knowledge interactive proofs.

1.1 Definitional issues: The original definition of zero-knowledge was presented in [GMR1]. This definition does not seem to fully capture the intuitive meaning of the concept of zero-knowledge. For one thing, one would expect that the sequential application ("composition") of protocols each of which is zero-knowledge would yield a protocol which is itself zero-knowledge (in the same manner that summing any finite number of zeros would leave the total at zero). However, as claimed by [FS] and recently shown in [GK], such a "composition theorem" cannot be proved for the [GMR1] definition. Another problem with this definition concerns its applicability to cryptographic protocols. Typically, zero-knowledge proof systems will be used as subprotocols within larger cryptographic protocols. In such a scenario it is natural that a dishonest party (a "cheating" verifier in the zero-knowledge terminology) will compute its messages based on information acquired *before* the proof protocol began, possibly from earlier stages of the protocol in which the zero-knowledge proof is a subprotocol. We would like to require that *even this additional information* will not enable the verifier to obtain any knowledge from its interaction with the prover. (This is not guaranteed by the original definition). In an effort to overcome these problems, we formulate the definition referred to as $auxiliary - input$ zero-knowledge. Intuitively, the definition requires that whatever a verifier that has access to any information can compute when interacting with the prover, it can also compute by itself when having access to the same information. Apart from dealing with verifiers that "cheat" by means of using outside information, the proposed definition also enables us to prove a composition theorem. The fact that auxiliary-input zero-knowledge is closed under composition is crucial for the use of zero-knowledge proofs in modular design of cryptographic protocols. In [GMW2] a compiler is presented that transforms any protocol correct in a weak adversarial model to a protocol correct in the strongest adversarial model. The existence of such a compiler relies heavily on the existence of *auxiliary-input* zero-knowledge proofs for every language in NP. On the other hand, the ability to derive such a strong result indicates that the auxiliary-input zero-knowledge definition is suitable for cryptographic purposes. The requirements of the auxiliary-input definition may seem very restrictive. However, all known zero-knowledge proof systems (e.g. [GMR1, GMW1]) satisfy even a seemingly much stricter definition.

All these protocols were proved zero-knowledge by presenting one algorithm that uses any verifier as a black-box to simulate the interaction of that verifier with the prover. In fact it is hard to conceive an alternative way to prove a protocol zero-knowledge. We therefore present the definition of $blackbox - simulation$ zero-knowledge, which formalizes this requirement. We show that blackbox-simulation zero-knowledge implies auxiliary-input zero-knowledge. As a result, all known zero-knowledge proofs are auxiliary-input zero-knowledge and can be used for cryptographic purposes such as those in [GMW2].

Remark 1.1: The fact that the [GMR1] definition is not closed under composition, and that "non-uniform" verifiers could be used to overcome this problem, was observed independently by Goldwasser, Micali and Rackoff [GMR2], Tompa and Woll [TW] and Feige and Shamir [FS].

1.2 Essential Properties of Zero-Knowledge: Other results in this paper concern the *triviality* of certain classes of zero-knowledge proof systems. We will consider a class of proof systems *trivial* in this context if only languages in BPP can have zero knowledge proof systems of this type. The reason being that any BPP language has a trivial zero-knowledge proof: one in which the verifier checks by himself whether $x \in L$ or not. Proving the triviality of some class of proof systems can be thought of as demonstrating that some property (which this class lacks) is essential to zero-knowledge. In particular, we show that any language L possessing a Las Vegas zero-knowledge proof system (i.e. a proof system that never causes the verifier to accept on $x \notin L$) is in Random Polynomial Time. It follows that the error probability on "no" instances, existing in all known zero-knowledge proofs, is inevitable and essential to the non-triviality of these proof systems. It is interesting to note that Las Vegas interactive proofs can exist only for languages in NP (see [GMS]). It is easy to see that the class of languages for which membership can be proved by a deterministic prover equals that for which membership can be proved by a probabilistic prover. (We can consider an *optimal* prover, i.e. one which always maximizes the acceptance probability. This prover computes in each case the "best possible" messages and can clearly be deterministic). Thus, randomness of the prover is not essential to the power of interactive proof systems as far as language recognition is concerned. On the other hand, in all proof systems shown to be zero-knowledge the prover is *probabilistic*, and this property seems essential to the "zero-knowledge-ness" of these proof systems. We show that this is no coincidence: only languages in BPP can have auxiliary-input zero-knowledge interactive proofs in which the prover is deterministic, and therefore randomness of the prover is essential to the non-triviality of the proof system. We thus demonstrate a meaningful difference between general interactive proofs and zero-knowledge interactive proofs. Just as an error probability on "no" instances and randomness both of the prover and the verifier are essential to zero-knowledge proof systems, so is the non-triviality of the interaction. It can be easily shown that only languages in BPP can have 1-step interactive proofs which are zero-knowledge. We show that the same holds for 2-step zero-knowledge proof systems under the auxiliary-input definition. Aiello and Hastad [AH2] proved that, relative to some oracle A, $2-$

$StepZero - Knowledge^A \not\subseteq BPP^A$. Their proof hold for the original [GMR1] definition (actually for a stronger definition, see [AH2]). The proof presents a 2-step protocol which is a zero-knowledge interactive proof system for some language L_A, but such that $L_A \notin BPP^A$. Since the prover in the protocol is deterministic, the result can also be interpreted as $Deterministic - ProverZero - Knowledge^A \not\subseteq BPP^A$. Our proofs for the 2-step and deterministic prover cases, both holding for auxiliary-input zero-knowledge, relativize, and we can therefore conclude that neither will extend to the [GMR1] definition. Note that 2-step protocols and deterministic-prover protocols *can* be zero-knowledge with respect to the prespecified verifier V (e.g. the 2-step protocols for Quadratic Non-Residuosity [GMR1] and Graph Non-Isomorphism [GMW1]). Therefore, unlike Fortnow [F] and Aiello and Hastad [AH1], who actually rely only on the fact that the prespecified verifier V has a simulator, we must in this case make use of the $full$ power of the definition of zero-knowledge: specifically, the requirement that there exist simulators for *all* verifiers, including the "cheaters". Our results extend to zero-knowledge arguments, introduced in [BCC]. Zero-knowledge arguments differ from zero-knowledge interactive proofs, which are the main topic of our investigations, in that the former have a relaxed soundness condition (rather than requiring that it is *impossible* to fool the verifier into believing false statements it is only required that cheating the verifier is *computationally infeasible*).

Remark 1.2: We stress that if one-way functions exist, then every language in $IP = PSPACE$ has a zero-knwoledge proof system [GMW1,IY,S]. These proof systems have all the essential properties discussed above. Hence there seems to be a big differnce between proof systems possesing these properties and those lacking them.

Organization of the Paper Section 2 contains some basic definitions and also an extension of the notion of *polynomial indistinguishability* which is required for the definitions presented in section 3. In section 3 we present our new definitions of zero-knowledge and investigate their relative power. We also prove the composition property of auxiliary-input zero-knowledge in that section. Section 4 contains our triviality results.

2. NOTATION AND BASIC DEFINITIONS Let S be a set. By $e \in_R S$ we mean that an element e is chosen at random from the set S with uniform probability distribution. When describing a protocol between two parties, A and B, we will write

$action$

to mean party A performs some internal action (computation), and

$A \rightarrow B$: m

to mean that A sends message m to B. We recall the definition of *interactive proof systems* [GMR1 ; An alternative definition due to Babai [B] was shown equivalent by Goldwasser and Sipser [GS]]: An interactive proof system for a language L is a protocol (i.e. a pair of local programs) for two probabilistic interactive machines called the *prover* and the *verifier*. Initially both machines have access to a common input tape. The two machines send messages to one another through two communication tapes. Each machine only sees its own tapes, the common input tape and the communication tapes. The verifier is bounded to a number of steps which is polynomial in the length of the common input, after which it stops in an *accept* state or in a *reject* state. We impose no restrictions on the local computation conducted by the prover. We require that, whenever the verifier is following its predetermined program, V, the following two conditions hold:

1) *Completeness of the interactive proof system:* If the prover runs its predetermined program, P, then, for every constant $c0$ and large enough $x \in L$, the verifier accepts the common input x with probability at least $1 - |x|^{-c}$. In other words, the prover can convince the verifier of $x \in L$.

2) *Soundness of the interactive proof system:* For every program P^*, run by the prover, for every constant $c0$ and large enough $x \notin L$, the verifier rejects x with probability at least $1 - |x|^{-c}$. In other words, the prover cannot fool the verifier.

An interactive proof system having P, V as programs will be denoted P, V.

Definition: A $t-$*step* interactive proof system is one in which a total of t messages is sent by the two parties. Without loss of generality, we assume that the last message sent during an interactive proof is sent by the prover. (A last message sent by the verifier can have no role in convincing the verifier and therefore has absolutely no effect.) Thus, the prover sends the last (and only) message in a 1-step interactive proof while in a 2-step protocol the verifier sends a message first, followed by a response from the prover. The notion of *polynomial indistinguishability* of probability distributions is used in the definitions of zero-knowledge discussed in the next section. We extend the original [GM, Y] definition for the case of probability distributions indexed by two parameters, which are treated differently. This extension is required for the formal definition of *auxiliary-input zero-knowledge*, presented in a later section. In that case, x will be the common input to the protocol while y will be the auxiliary input to the verifier.

Definition (polynomial indistinguishability): For every algorithm A, let $p_A^{D(x,y)}$ denote the probability that A outputs 1 on input (x,y), an element chosen according to the probability distribution $D(x,y)$. Denote by Dom the domain from which the pairs x, y are chosen. The distribution ensembles $\{D(x,y)\}_{x;y \in Dom}$ and $\{D\prime(x,y)\}_{x;y \in Dom}$ are *polynomially indistinguishable* if for every probabilistic algorithm A which runs time polynomial in the length of its input, for every constant $c0$ there exists N_0 such that for every x, $|x| > N_0$, and for every y such that $(x,y) \in Dom$,

$$|p_A^{D(x,y)} - p_A^{D\prime(x,y)}| \leq |x|^{-c}.$$

Note that we do not put any restrictions on the length of y, and in particular we don't require $|y| N_0$. The original definition is obtained from the above definition by omitting all mention of y. We will occasionally avoid specifying the domain, and write $\{D(x,y)\}_{x;y}$ instead of $\{D(x,y)\}_{x;y\in Dom}$. Two distribution ensembles $\{D(x,y)\}_{x;y\in Dom}$ and $\{D\prime(x,y)\}_{x;y\in Dom}$ are NOT polynomially indistinguishable if there exist a probabilistic polynomial time algorithm A , a constant $c0$, and an infinite sequence Seq of x's such that for every $x \in Seq$ there exist some y such that $(x,y) \in Dom$ and

$$p_A^{D(x,y)} - p_A^{D\prime(x,y)} \geq |x|^{-c}.$$

Definition: Let $c0$ be a constant and let $D(x,y)$ and $D\prime(x,y)$ be probability distributions over strings of length $n1$. We say that an algorithm A *c-distinguishes* between $D(x,y)$ and $D\prime(x,y)$ if

$$p_A^{D(x,y)} - p_A^{D\prime(x,y)} \frac{1}{n^c}$$

Remark 2.1: Throughout this paper we use the phrases "with very high probability", "with (non-) negligible probability", and so on, to describe the behavior of algorithms. The formal interpretation of the statement "the algorithm behaves this way with very high probability" should be taken to be "the probability that the algorithm behaves this way on input of length n is greater than $1 - \frac{1}{Q}(n)$ for any (positive) polynomial Q and sufficiently large n". Accordingly "negligible probability" is "less than $\frac{1}{Q}(n)$ for any (positive) polynomial Q and sufficiently large n" and "non-negligible probability" means "greater than $\frac{1}{Q}(n)$ for some polynomial Q and sufficiently large n". For convenience, we will say that a function $p(n)$ is *c-non-negligible*, where $c0$, if $p(n)\frac{1}{n^c}$ for infinitely many n's.

3. A TAXONOMY OF ZERO-KNOWLEDGE DEFINITIONS In this section we present two alternative definitions of the notion of zero-knowledge, and investigate the relationship between them. We start by defining *history descriptions* and recalling the original zero-knowledge definition of [GMR1].

Definition: A *history description* of a conversation between a machine V^* and the prover P consists of the contents of all of V^*'s read-only tapes (common input, random input, and in the case of auxiliary-input zero-knowledge, also the auxiliary input) and of the sequence of messages sent by the prover during the interaction. We use $[x,r,m]$ ($[x,y,r,m]$) to denote history descriptions, where x is the common input, (y the auxiliary input), r the random input to the verifier and m the sequence of messages sent by the prover. We denote by $P(x), V^*(x)$ ($P(x), V^*(x,y)$) the probability distribution of history descriptions generated by the interaction of V^* with P on $x \in L$.

Definition: [GMR1]: An interactive proof system for a language L is *zero-knowledge* if for all probabilistic polynomial-time machines V^*, there exists a probabilistic polynomial-time algorithm M_{V^*} that on input x produces a probability distribution $M_{V^*}(x)$ such that $\mathrm{M_{V^*}(x)}_{x \in L}$ and $\mathrm{P(x), V^*(x)}_{x \in L}$ are polynomially indistinguishable.

Remark 3.1: If we require that the above two probability distributions be *equal*, we obtain the definition referred to as *perfect zero-knowledge*. If we require them to be statisticly close, we obtain *almost-perfect zero-knowledge*. (The definitions originate from [GMR1], and were named as above in [F].)

Remark 3.2: In the definition above we required M_{V^*} to simulate the *history* of V^*'s interaction with P. An alternative definition is to require M_{V^*} to generate the *output* of V^* when interacting with P. Clearly, the *output* of V^* is determined given the *history*, and therefore simulating the history is at least as hard as simulating the output. The converse may not be true for a specific verifier (in particular for V, the "honest" verifier). However, since for every verifier V^* there exists a verifier $V\prime$ whose output is the *history* of the interaction of V^* with P, it follows that, when quantifying over all verifiers, the two formalizations are equivalent. We will be using the history-based notion of zero-knowledge throughout this paper.

3.1 New Definitions The first definition to be considered is motivated by cryptographic applications and will be referred to as the *Auxiliary Input* Zero-Knowledge definition. Let us elaborate on this motivation: Zero-knowledge interactive proofs are a powerful tool in the design of cryptographic protocols. Typically, they will be used by a party to prove that it is computing its messages according to the protocol. It is crucial that these proofs are carried out without yielding the prover's secrets. In such a scenario it seems natural to assume that an adversarial party playing the role of the "verifier" will try to gain knowledge of interest to it. In order to do so the adversary may deviate from the specified program and compute its messages in a manner suited to its goals. Most probably it will want to base the computation of its messages on previously acquired information, possibly from earlier stages of the protocol in which the zero-knowledge proof is a subprotocol. Intuitively, we will require that the proof system be such that even having this additional information cannot enable any V^* to extract from its conversations with P anything that it could not compute by itself having that same information. To allow this possibility the interactive proof and zero-knowledge definitions introduced in [GMR1] should be modified so that the verifier can have an auxiliary input tape, through which the information that enables the "verifier" to compute the desired messages will be entered.

Definition (Auxiliary-input Zero-Knowledge): An interactive proof system for a language L is *auxiliary-input zero-knowledge* if for every probabilistic polynomial time machine V^* there exists a probabilistic polynomial time machine M_{V^*} such that the distribution ensembles $\{P(x), V^*(x,y)\}_{x;y \in D_1}$ and $\{M_{V^*}(x,y)\}_{x;y \in D_1}$ are polynomially indistinguishable, where $D_1 = \{(x,y) | x \in L, y \in \{0,1\}^*\}$. Note that by saying that V^* is polynomial-time we mean that its running time is bounded by a polynomial in the length of the common input. Machine V^* has an additional input tape containing the auxiliary input y. During an interaction of V^* on common input x, machine V^* reads at most a $poly(x)$-long prefix of its auxiliary input. A similar convention holds for the simulator M_{V^*} (i.e. its running time is polynomial in the length of its first input, and consequently it may only read a prefix of the second input. The second definition we consider will be referred to as *Blackbox Simulation Zero-Knowledge*. This definition requires the existence of a single polynomial time machine M_u which simulates the interaction of *any* polynomial time machine V^* with the prover P on any $x \in L$, using V^* as a blackbox. What do we mean by "use V^* as a blackbox"? A probabilistic algorithm in general can be viewed either as an algorithm which internally tosses coins or as a *deterministic* algorithm that has two inputs: a regular input and a random input. Two corresponding interpretations of "using a probabilistic algorithm as a blackbox" follow. In the first case, it means choosing an input and running the algorithm, while the algorithm internally flips its coins. In the second case, it means choosing both inputs, and running the algorithm (the second input serves as the outcome of random coin tosses). Both these approaches extend naturally to probabilistic algorithms which also interact with other machines, as in our case. We choose to adopt the second approach, that is, when using V^* as a blackbox, the simulator M_u will choose both inputs to V^*. All known zero-knowledge protocols were proved zero-knowledge using this approach. It is not clear if they could also be proved zero-knowledge when adopting the first approach.

Definition (Black-box Simulation Zero-Knowledge): Denote by $Time_P^{V^*}(x)$ the running time of machine V^* when interacting with P on input x. An interactive proof system for a language L is *black-box simulation zero-knowledge* if there exists a probabilistic polynomial-time machine M_u such that for every polynomial Q the distribution ensembles $\{P(x), V^*(x)\}_{x;V^* \in D_2}$ and $\{M_u^{V^*}(x)\}_{x;V^* \in D_2}$ are polynomially indistinguishable even when the distinguishers are allowed blackbox access to V^*, where $D_2 = \{(x, V^*) | x \in L \ and \ Time_P^{V^*}(x) \leq Q(x)\}$.

All known zero-knowledge protocols are in fact blackbox-simulation zero-knowledge. It seems likely that in order to prove an interactive proof system zero-knowledge with respect to *any* "verifier" V^*, one would have to present such a universal simulator. Thus this definition is reasonable and not too restrictive.

Remark 3.3: In remark 3.2 of this section we claimed that the "history-based" and the "output-based" versions of the [GMR1] zero-knowledge definitions are equivalent. This claim was established by pointing out that the distinguisher, given a history description, can generate V^*'s output by using a built-in version of V^*. The same reasoning holds for the auxiliary-input definition. However the distinguishers in the case of blackbox-simulation cannot have a built-in version of what may be an infinite number of V^*'s. Therefore one must allow the distinguishers running on a history description of an interaction by some machine V^* blackbox access to V^*. This will clearly allow the distinguisher to reconstruct V^*'s output given the history of the interaction.

Remark 3.4: We stress that saying that (P, V) is an auxiliary input zero-knowledge proof system does not mean that the honest verifier V may use auxiliary input as a legitimate stage in its operation (it may not). Rather, we mean that the prover does not reveal knowledge even to cheating verifiers which do use an auxiliary input.

3.2 Relationship Between the Definitions Let $Cl(def)$ denote the class of all interactive proof systems satisfying the requirements of definition def. The following relationships seem rather obvious:

Theorem 3.1:

(1) $Cl(auxiliary - input) \subset Cl([\text{GMR1}])$

(2) $Cl(blackbox - simulation) \subset Cl([\text{GMR1}])$

Proof: In both case (1) and case (2) the [GMR1] definition is less restrictive than the other definitions in terms of its requirements from the simulation. In case (1) the simulation is required by the [GMR1] definition to be valid only when the auxiliary-input is empty. In case (2) the blackbox definition requires that all verifiers be simulated by one machine M_u whereas the [GMR1] definition allows each such verifier to have its own, specially tailored simulator. □

Next, we establish the relationship between the two new definitions:

Theorem 3.2: $Cl(blackbox - simulation) \subset Cl(auxiliary - input)$.

Proof: Let P, V be an interactive proof system and assume $P, V \in Cl(blackbox - simulation)$. That is, there exists a polynomial machine M_u such that for every $x \in L$ and $V\prime$ machine M_u simulates the interaction of $V\prime$ with P on input x. We show $P, V \in Cl(auxilary - input)$, by demonstrating how to construct a simulator M_{V^*} for every probabilistic polynomial-time V^* having auxiliary input. For every V^* we construct M_{V^*} as follows: Let Q be a polynomial such that $\forall x Time_P^{V^*}(x) \leq Q(x)$. The simulator M_{V^*} will be a multiple-tape Turing machine. It will have the code of V^* built-in. M_{V^*} will also have access to M_u, the universal simulator guaranteed by the blackbox definition. Given x and auxiliary-input y, machine M_{V^*} "incorporates" a prefix of y of length $\leq Q(x)$ into the code of V^*, forming a machine V_y^*. On input x, machine V_y^* behaves as follows: it copies y to its input tape and runs $V^*(x, y)$. Also, upon receiving a message "SEND AUXILIARY INPUT", V_y^* sends a message containing y (this feature is not required by the simulation, but will later be used by the distinguishers). Having constructed V_y^*, machine M_{V^*} now simulates the computation of M_u while having the "blackbox" V_y^*. It then outputs the output of M_u. Observe that the output of M_u will be of the form $[x, r, m]$ while the output of M_{V^*} must be of the form $[x, y, r, m]$. Therefore M_{V^*} adds y to the output of M_u.

Claim 3.2.1: $M_{V^*}(x,y)$ runs time polynomial in $|x|$, as required by the definition of auxiliary-input zero-knowledge.

Proof: The time required to simulate one step of $V_y^*(x,y)$ is $O(|V^*|+|y|)$. The value of $|V^*|$ is constant as far as M_{V^*} is concerned, and therefore one step requires $O(|y|)$. Since y was truncated to length $Q(|x|)$, it follows that $|y| \in O(Q(|x|))$. We know that $V_y^*(x)$, which is essentially the same as $V^*(x,y)$, runs at most $Q(|x|)$ steps. All in all, simulating the computation of $V_y^*(x)$ can be achieved in time bounded by some polynomial $Q_V(|x|)$. M_{V^*} simulates the computation of M_u having a blackbox V_y^*. The number of steps required by M_u is guaranteed to be polynomial in $|x|$, when counting the activations of the blackbox V_y^* at unit cost. Let $Q_M(|x|)$ be the running time of M_u. The running time of M_{V^*} is bounded by $Q_M(|x|) \cdot Q_V(|x|)$ and is clearly polynomial in $|x|$. $\Box$

Claim 3.2.2: The distribution ensemble $\{P(x), V^*(x,y)\}_{x;y\in D_1}$ is polynomially indistinguishable from $\{M_{V^*}(x,y)\}_{x;y\in D_1}$, where $D_1 = \{(x,y)|x \in L\}$.

Proof: Assume there exist a constant c, an algorithm A and an infinite sequence S of pairs $(x,y) \in D_1$ such that

$$\forall\,(x,y) \in S : p_A^{P(x),V^*(x,y)} - p_A^{M_{V^*}(x,y)} \frac{1}{|x|^c}$$

We show that in such a case there exist a polynomial Q, an algorithm $A\prime$ and an infinite sequence $S\prime$ of pairs (x, V_y^*) such that

$$S\prime \subset \{(x,V_y^*)\,|\,x \in L\,,\,Time_P^{V_y^*}(x) \leq Q(|x|)\} and \forall\,(x,V_y^*) \in S\prime p_{A\prime}^{P(x),V_y^*(x)} - p_{A\prime}^{M_u^{V_y^*}(x)} \frac{1}{|x|^c}$$

contrary to the assumption that M_u is a valid blackbox simulator. Let $S\prime = \{(x,V_y^*)\,|\,(x,y) \in S\}$, where V_y^* is as described above. Clearly

$$\forall\,(x,V_y^*) \in S\prime Time_P^{V_y^*}(x) \;=\; Q(|x|).$$

We construct $A\prime$, the "blackbox-simulation" distinguisher, as follows: On input $[x,r,m]$ and a blackbox V_y^* (recall that blackbox distinguishers have blackbox access to the verifiers), $A\prime$ first sends a message "SEND AUXILIARY INPUT" to V_y^*, to obtain y. It then runs $A([x,y,r,m])$ and outputs the outcome of this computation. It is easy to see that $A\prime$ will distinguish for any pair (x,V_y^*) for which A distinguishes the corresponding pair (x,y). The claim follows. $\Box$ This completes the proof of Theorem 3.2. $\Box$

This is the most important result of this section, due to its effect: all known zero-knowledge protocols, having been proved zero-knowledge under the blackbox simulation definition, are shown to be auxiliary-input zero-knowledge, and as such can be used for all cryptographic applications such as [GMW2].

Remark 3.5: The relationships derived in the above theorems hold also for *perfect zero-knowledge* and *almost-perfect zero-knowledge.*

Remark 3.6: It follows from [GK, Thm. 4.1] that $Cl(auxiliary - input) \subset Cl([\mathrm{GMR1}])$. We don't know whether $Cl(auxiliary - input)$ equals $Cl(blackbox - simulation)$. The following states clearly what is known:

$$Cl(blackbox - simulation) \subset Cl(auxiliary - input) \subset Cl([\mathrm{GMR1}]).$$

3.3 Proof of the Sequential Composition Theorem for Auxiliary-Input Zero-Knowledge We first define the notion of a sequential composition of interactive proof systems:

Definition: Let P_1, V_1, ... , P_k, V_k be interactive proof systems for languages L_1, L_2, ...,L_k, respectively. A *sequential composition* of the k protocols, denoted P, V is defined as follows: The common input to P, V, x, will be a string of the form $x_1\%x_2\%...\%x_k\%$, where '%' is a delimiter. The execution of P, V consists, at stage i, of P and V activating P_i and V_i, respectively, as subroutines on x_i. V accepts if all V_i's have accepted.

In a similar manner we can define concurrent compositions:

Definition: Let P_1, V_1, ... , P_k, V_k be interactive proof systems for languages L_1, L_2, ...,L_k, respectively. Without loss of generality, assume that all protocols are m-step protocols. A *concurrent composition* of the k protocols, P, V is defined as follows: P, V will also be an m-step protocol. The common input to P, V, x, will be a string of the form $x_1\%x_2\%...\%x_k\%$, where '%' is a delimiter. The i'th message in P, V will consist of the i'th message of P_1, V_1, ... , P_k, V_k. V accepts if all V_i's have accepted.

Remark 3.7: Clearly, the case in which a single protocol $\hat{P}, \hat{V}$ is iterated k times, possibly on the same input $\hat{x}$, is merely a restricted version of the above definitions, in which $\forall i P_i, V_i = \hat{P}, \hat{V}$ and $\forall i x_i = \hat{x}$. It is easy to see that both compositions (sequential and concurrent) constitute interactive proofs for L. We now prove that a *sequential* composition of auxiliary-input zero-knowledge protocols yields a auxiliary-input zero-knowledge protocol. Recently it was shown in [GK] that the same is not true for concurrent compositions.

Remark 3.8: In the following proofs k, the number of protocols, is assumed to be constant. We will later demonstrate how a slightly altered version of the proof can be applied in the meaningful cases for which k is not a constant.

Theorem 3.3 (Sequential Composition Theorem): Let P_1, V_1, P_2, V_2, ... ,P_k, V_k be auxiliary-input zero-knowledge proof systems for languages L_1, L_2, ... , L_k, respectively. Let $L = \{x_1\%x_2\%...\%x_k\% | \forall i(x_i \in L_i)\}$. Define P, V to be the composition of P_1, V_1, P_2, V_2, ... , P_k, V_k. Then P, V is an auxiliary-input zero-knowledge proof system for L.

Proof: It is easy to see that P, V is an interactive proof system for L. We therefore concentrate on showing that P, V is auxiliary-input zero-knowledge. Recall that we are using the history-based notion of zero-knowledge. A history description in the case of auxiliary-input is of the form $[x, y, r, m]$, where y is the verifier's auxiliary input, r is the verifier's random string and m is the sequence of prover messages.

The objective of the indented small-print paragraphs throughout the proof is to provide insight and intuition to the otherwise rather formal proof.

In order to prove that P, V is auxiliary-input zero-knowledge we must show how to construct a simulator M_{V^*} for each polynomial-time probabilistic V^*. We will assume without loss of generality that V^* initially copies the contents of all its input tapes (common input, random input, auxiliary input) to its work tape and never attempts to access these tapes again.

V^*'s interaction with P can be conceptually divided into V^*'s interaction with P_1, V^*'s interaction with P_2, and so on. Since the k individual protocols are auxiliary-input zero-knowledge, there must exist machines $M^1_{V^*}$, $M^2_{V^*}$, ...,$M^k_{V^*}$, which simulate the interaction of V^* with P_1, P_2, ..., P_k, respectively. Basicly, M_{V^*} will activate these simulators in sequence. However, in order for the overall simulation to be valid, the initial state of V^* when being simulated by $M^{i+1}_{V^*}$ should be its final state in the simulation by $M^i_{V^*}$. This can be achieved by giving V^* as its auxiliary input to the $i+1$-th stage information which will enable it to reconstruct the final state of the i-th stage. Obviously, we cannot guarantee that any V^* will in fact behave as described above (i.e. reconstruct its state when having past history as its auxiliary input). Therefore, and instead of making any technical assumptions on V^*, we consider for every V^* a modified verifier $V\prime$ which will exhibit the required behavior.

As a first step we will consider a verifier $V\prime$ that has a built-in version of V^* and the following additional property: On auxiliary input h, where $h = [x, y, r, m]$ is a history description of V^*'s interaction with the prover, $V\prime$ brings its built-in version of V^* to the configuration (state, work-tape contents and head position) corresponding to this description, and proceeds from that point. In particular, if $m = \epsilon$ (the empty string) and y is not itself a history description then $V\prime$ only copies x, y, r to the work tape of its built-in version of V^* and then "behaves" like V^*. Machine $V\prime$ actually always ignores its "real" random string. In all other senses $V\prime$ is exactly like V^*. In particular, for every x, y, the probability distribution of prover messages sequences generated by running $P(x), V^*(x, y)$ is exactly that generated by randomly choosing a string r and running $P(x), V\prime(x, [x, y, r, \epsilon])$.

Construction of the simulator for V^*: Since the individual protocols are assumed to be auxiliary-input zero-knowledge, there exists machines $M^1_{V\prime}$, $M^2_{V\prime}$, ... , $M^k_{V\prime}$ which simulate the history of V''s interaction with P_1 , P_2, ...,P_k , respectively. The output produced by $M^i_{V'}$ on input pair (x, h) will be of the form $[x, h, r, m]$, where r is V''s random string (which is actually ignored) in this simulation and m is the sequence of messages sent "on behalf" of the prover. Let $s_1 s_2$ denote the concatenation of strings s_1 and s_2. We now describe M_{V^*}. On input $x = x_1\%x_2\% \cdots x_k\%$ and y, machine M_{V^*} runs

choose random string r
$h_0 - [x, y, r, \epsilon]$
$h_1 - M^1_{V'}(x_1, h_0)$
$h_2 - M^2_{V'}(x_2, h_1)$
...
$h_k - M^k_{V'}(x_k, h_{k-1})$
$m - m_1 m_2 ... m_k$
$OUTPUT([x, y, r, m])$.

(The m_i's are obtained from the h_i's.) We will now show that M_{V^*} is indeed a "good" simulator for P, V^*.

Lemma 3.3.1: The distribution ensembles $\{M_{V^*}(x, y)\}_{x,y}$, where M_{V^*} is as described above, and $\{P(x), V^*(x, y)\}_{x,y}$ are polynomially indistinguishable.

Proof: Suppose they are not. That is, there exists a constant $c0$ and a test A that for infinitely many pairs (x, y) will c-distinguish between $M_{V^*}(x, y)$ and $P(x), V^*(x, y)$

We will show that in such a case there exists another constant c' and another test $A^{(i)}$ that for some i and for infinitely many pairs (x_i, y_i) c'-distinguishes between $M^i_{V'}(x_i, y_i)$ and $P_i(x_i), V'(x_i, y_i)$, contrary to the assumption that $M^i_{V'}$ correctly simulates the history of V''s interaction with P_i.

We consider the following *hybrids* of the probability distributions $M_{V^*}(x, y)$ and $P(x), V^*(x, y)$. The i-th hybrid, denoted $H_i(x, y)$, is defined by the following process:

choose a random string r
$h_0 - [x, y, r, \epsilon]$
$h_1 - P_1(x_1), V'(x_1, h_0)$
$h_2 - P_2(x_2), V'(x_2, h_1)$
...
$h_i - P_i(x_i), V'(x_i, h_{i-1})$
$h_{i+1} - M^{i+1}_{V'}(x_{i+1}, h_i)$
...
$h_k - M^k_{V'}(x_k, h_{k-1})$,
$m - m_1 m_2 ... m_k$
$OUTPUT([x, y, r, m])$.

As before, each h_i is of the form $[x, h_{i-1}, r_i, m_i]$. The extreme hybrids , H_0 and H_k, correspond to $M_{V^*}(x,y)$ and $P(x), V^*(x,y)$, respectively. Clearly if we can c-distinguish between the extreme hybrids, then there must exist a constant $c\prime$ and two adjacent hybrids which can be $c\prime$-distinguished, say H_{i-1} and H_i. It is not hard to see that for sufficiently large n, $c\prime$ is approximately equal to c. Let $pref_i(x,y)$ be the probability distribution defined by the process

$choose\ a\ random\ string\ r$
$h_0 - [x,y,r,\epsilon]$
$h_1 - P_1(x_1), V\prime(x_1,h_0)$
$h_2 - P_2(x_2), V\prime(x_2,h_1)$
...
$h_{i-1} - P_{i-1}(x_{i-1}), V\prime(x_{i-1},h_{i-2})$
$OUTPUT(h_{i-1})$.

Let h be a string which may occur with non-zero probability in either of the distributions $M^i_{V\prime}(x_i,h_{i-1})$, and $P_i(x_i), V\prime(x_i,h_{i-1})$, where h_{i-1} is a string assigned non-zero probability by $pref_i(x,y)$. Any such string h will contain m_1, m_2, ..., m_i and x, y and r. For strings h of this type we define $suff_i(h)$ to be the probability distribution generated by running

$h_{i+1} - M^{i+1}_{V\prime}(x_{i+1},h)$
$h_{i+2} - M^{i+2}_{V\prime}(x_{i+2},h_{i+1})$
...
$h_k - M^k_{V\prime}(x_k,h_{k-1})$
$m - m_1m_2...m_k$
$OUTPUT([x,y,r,m])$.

The distribution $pref_i(x,y)$ is actually a distribution on all the possible auxiliary inputs to the i-th stage, given that the initial input is x and the initial auxiliary input is the string y. The distribution $suff_i$ can be regarded as an operator which on input a stage i history applies the remaining $k-i$ simulation stages. If the input to $suff_i$ comes from $M^i_{V\prime}(x_i,h_{i-1})$ then the effect of $suff_i$ will correspond to a string coming from $H_{i-1}(x,y)$. If the input comes from $P_i(x_i), V\prime(x_i,h_{i-1})$, then the effect of $suff_i$ will correspond to a string coming from $H_i(x,y)$. Our aim is to show that if (x,y) are such that A c-distinguishes between $H_0(x,y)$ and $H_k(x,y)$ then there exists some i and some h^* such that the $A^{(i)}$ we construct will $c\prime$-distinguish between between $M^i_{V\prime}(x_i,h^*)$ and $P_i(x_i), V\prime(x_i,h^*)$. $A^{(i)}$ will actually activate the $suff_i$ operator on its input text, h, to obtain a text in a format suitable for A, and then "let A do the distinguishing".

We use the following notational shorthands:

$PR_i[h] = Prob\{pref_i(x,y) = h\}$
$suff_i(M[h]) = suff_i(M^i_{V\prime}(x_i,h))$
$suff_i(P[h]) = suff_i(P_i(x_i), V\prime(x_i,h))$

Recall that p^D_A denotes the probability that algorithm A outputs 1 on input of an element chosen according to the probability distribution D. The following relationship holds:

$$p_A^{H_{i-1}(x,y)} = \sum_h PR_i[h] \cdot p_A^{suff_i(M[h])}$$

The probability $p_A^{H_{i-1}(x,y)}$ is written above as a weighted average over all the possible h's, of the probability that A outputs 1 on input an element chosen according to $suff_i(M[h])$. The weight is assigned by the probability of h to be an $i-1$ stage history.

Similarly:

$$p_A^{H_i(x,y)} = \sum_h PR_i[h] \cdot p_A^{suff_i(P[h])}$$

It was assumed that the values $p_A^{H_{i-1}(x,y)}$ and $p_A^{H_i(x,y)}$ differ c'-non-negligibly. Since both are weighted averages over the same probability space, there must be some element h^* for which there will be a c'-non-negligible difference between $p_A^{suff_i(M[h^*])}$ and $p_A^{suff_i(P[h^*])}$.

Since $p_A^{H_{i-1}(x,y)} - p_A^{H_i(x,y)} > / < \frac{1}{|x|^{c'}}$ there exists some h^* for which

$$p_A^{suff_i(M[h^*])} - p_A^{suff_i(P[h^*])} \frac{1}{|x|^{c'}}$$

We conclude that for every (x,y) for which $H_0(x,y)$ and $H_k(x,y)$ can be c-distinguished there exists (x_i,y_i) such that $P_i(x_i), V'(x_i,y_i)$ and $M^i_{V'}(x_i,y_i)$ can be c'-distinguished. The auxiliary input y_i will be the string h^* corresponding to x and y. On input a text $T = [x_i, y_i, r_i, m_i]$ chosen either according to $P_i(x_i), V'(x_i,y_i)$ or to $M^i_{V'}(x_i,y_i)$, the test $A^{(i)}$ extracts m_1, m_2, ...,m_{i-1}, x , y and r (which are contained in T since they were contains in $y_i = h^*$) and m_i from T. It then runs

$$\begin{array}{l} h_{i+1} - M^{i+1}_{V'}(x_{i+1}, T) \\ h_{i+2} - M^{i+2}_{V'}(x_{i+2}, h_{i+1}) \\ \ldots \\ h_k - M^k_{V'}(x_k, h_{k-1}) \\ m - m_1 m_2 \ldots m_k \\ OUTPUT([x,y,r,m]) \end{array}$$

to obtain a text $T' = [x,y,r,m]$. The test $A^{(i)}$ then runs A on T' and outputs the output of A. By our construction it is clear that $A^{(i)}(z)$ will c'-distinguish between $P_i(x_i), V'(x_i,y_i)$ and $M^i_{V'}(x_i,y_i)$. This contradicts the fact the $M^i_{V'}$ is a "good" simulator for P_i, V'. $\Box$ We conclude that $\{M_{V^*}(x,y)\}_{x;y}$ is polynomially indistinguishable from $\{P(x), V^*(x,y)\}_{x;y}$ and the theorem follows. $\Box$

Remark 3.9: The assumption that k, the number of protocols, is constant was required in order to argue that if H_0 and H_k can be distinguished for infinitely many pairs (x,y) then there exists some i such that H_{i-1} and H_i can also be distinguished infinitely many times, thus contradicting the assumption that $M^i_{V'}$ is a good simulator. Observe, however, that in the case where a single protocol $\hat{P}, \hat{V}$ is iterated, it is no longer essential to assume that k is a constant. Clearly we could no longer claim that for some i the distributions H_{i-1} and H_i can be distinguished infinitely many times. However, distinguishing any two adjacent hybrids H_{i-1} and H_i means in every case distinguishing $M_{V\hat{\prime}}$ from $\hat{P}, V\hat{\prime}$, contrary to the assumption that $M_{V\hat{\prime}}$ is a good simulator for $\hat{P}, V\hat{\prime}$. Therefore the Sequential Composition Theorem holds also in this case. More generally, the Sequential Composition Theorem holds for non-constant k whenever in each of the k stages one of a finite set of protocol is run.

Remark 3.10: An analogous Sequential Composition Theorem can be proved for the blackbox-simulation zero-knowledge definition.

4. ESSENTIAL PROPERTIES OF ZERO-KNOWLEDGE PROOFS In this section we show that certain properties are essential to zero-knowledge proof systems. We do so by demonstrating the *triviality* of zero-knowledge proof systems lacking these properties. By a *class* of interactive proof systems we mean, for example, all proof systems in which the verifier is deterministic, all proof systems in which only one message is sent, and so on. Let us first discuss the meaning of triviality in this context. The complexity class BPP encompasses our notion of efficient computation. Recall that a language L is in BPP if there exists a probabilistic polynomial time machine M such that for every constant $c0$ and large enough x,

$$\text{if } x \in L \ Prob\,(M(x) = ACC) \geq 1 - |x|^{-c} \ (\textit{Completeness} \text{ condition})$$

$$\text{if } x \notin L \ Prob\,(M(x) = REJ) \geq 1 - |x|^{-c} \ (\textit{Soundness} \text{ condition})$$

Since V can recognize by itself any language in BPP, it follows that any language in BPP has a *trivial* zero-knowledge proof system: one in which the verifier checks by itself if $x \in L$ or not. Accordingly, we consider any class of zero-knowledge interactive proofs *trivial* if proof systems of this class can be zero-knowledge only for languages in BPP.

4.1 General Framework of Triviality Proofs Basicly, our proof method will be the following: to prove the triviality of some class C, we will assume that some language L has a zero-knowledge proof system of class C. By the definition of zero-knowledge there exists a simulator M_V which generates history descriptions of the interaction of V with the prover P (in some cases we will consider the simulator with respect to some cheating verifier V^*, that is M_{V^*}). We will build a BPP machine for L, that uses M_V (M_{V^*}). Let $H = [x, r, m]$ be a history description ($H = [x, y, r, m]$ in the case of auxiliary-input), where x is the common input, (y is the auxiliary input), r is the random input and m is the sequence of messages sent in the protocol. The string m is of the form $(\alpha_0, \beta_1, \cdots, \alpha_k)$ where the α's are the prover messages and the β's are the verifier messages (m will be of the form $(\beta_1, \alpha_1, \cdots, \alpha_k)$ if in the protocol V "speaks" first). We will denote by $V^*(x, r, \alpha_0, \cdots, \alpha_{i-1})$ the deterministic polynomial-time computation that a verifier V^* uses to determine β_i (in the case of auxiliary input $\beta_i = V^*(x, y, r, \alpha_0, \cdots, \alpha_{i-1})$). Similarly, $P(x, \beta_1, \cdots, \beta_i)$ will denote the probabilistic computation used by P to determine α_i. The computation used by the honest verifier, V, to determine whether to accept or to reject will be denoted $\rho(x, r, \alpha_1, \cdots, \alpha_k)$.

Definition: A history description (or "conversation") $H = [x, r, m]$ ($H = [x, y, r, m]$ in the case of auxiliary-input), is *legal with respect to a verifier* V^* if the messages contained in m satisfy the following requirement:

$$\forall i 1 \leq i \leq k \beta_i = V^*(x, r, \alpha_0, \cdots, \alpha_{i-1})$$

(In the case of auxiliary-input: $\beta_i = V^*(x, y, r, \alpha_0, \cdots, \alpha_{i-1})$). For convenience, we will simply say "H is legal" when the identity of V^* is clear from the context. H is *accepting* if it is legal with respect to V and if

$$\rho(x, r, \alpha_0, ..., \alpha_k) = ACC$$

Accepting conversations are only defined with respect to V. Recall that the texts produced by M_V on input $x \in L$ must be polynomially indistinguishable from the texts of real interaction between V and P. Therefore, and since a real conversation between P and V on $x \in L$ will be with very high probability legal and accepting, it follows that M_V must also produce legal and accepting conversations with very high probability for $x \in L$, and do so within polynomial time. Otherwise a distinguisher which simply outputs 1 if the given conversation is accepting will clearly distinguish between real interactions and simulation texts. The definition(s) of zero-knowledge require nothing of M_V in the case $x \notin L$. The result of running M_V on $x \notin L$ may be one of the following:

1) M_V may run for too long.

2) M_V may produce a non-accepting (though perhaps legal) conversation.

3) M_V may produce an accepting conversation.

The third case is indeed possible: in all protocols demonstrated to be zero-knowledge (e.g. [GMR1, GMW1]) the simulator presented in the proof generates accepting conversations regardless of whether x is in the language or not. In fact, if this case were not possible, then for any language which has a zero-knowledge proof system we could easily build a BPP machine: the machine would run M_V on x and accept if and only if M_V produces an accepting conversation. We conclude that a BPP machine which runs M_V can "safely" reject if either case 1 or case 2 occurs, because they are guaranteed to occur with negligible probability for $x \in L$. The hard case to handle is the third case. In the proofs throughout this section we will for each instance use the special structure of the specific class of interactive proofs under consideration to handle this case. While using M_V (M_{V^*}) in the proofs that follow we will usually claim that some property, existing in the texts of real interaction on $x \in L$, must also exist with very high probability in the texts produced by the simulator on input $x \in L$ (For example, a property such as "the text constitutes an accepting conversation"). If the protocol is perfect or almost-perfect zero-knowledge, this claim follows immediately. However, if the two probability distributions are "only" polynomially-indistinguishable (following [AH1], we will refer to this case as *computational* zero-knowledge), the proof may become more involved. In each case we will first present a proof for perfect zero-knowledge, and then adapt it to computational zero-knowledge. Each formal proof will be preceded by an intuitive discussion of the main ideas underlying it.

Remark 4.1: In the proofs that follow the BPP machines built are actually shown to satisfy the requirements of BPP for all but *perhaps a finite set of x's.* Clearly any such machine can be transformed into a "true" BPP machine.

4.2 Zero-Knowledge Proofs Which Never Err and Zero-Knowledge Proofs with Deterministic Verifiers M. Blum proposed the concept of "Las Vegas" interactive proofs. Informally, these are interactive proof systems that never err, that is never cause V to accept when $x \notin L$. In [GMS] these protocols are referred to as "interactive proofs with *perfect soundness*". In this section we show that no protocol of this type can be zero-knowledge, even with respect to the [GMR1] definition. A formal definition of "Las-Vegas Interactive Proofs" can be obtained from the definition of general interactive proofs simply by replacing the soundness condition with: "whenever $x \notin L$, and for every program P^* run by the prover, either V rejects or the protocol does not terminate".

Theorem 4.1: Let L be a language for which there exists a zero-knowledge Las Vegas interactive proof system. Then $L \in RP$.

Proof: The idea is to show that in this case accepting conversations simply do not exist for $x \notin L$, while (as always) for $x \in L$ the simulator M_V will produce accepting conversations with very high probability. Let us first recall the definition of *Random Polynomial Time*: A language L is in RP if there exists a probabilistic polynomial time algorithm M such that

on input $x \in L$ machine M accepts with probability $> 1/2$ (completeness)
on input $x \notin L$ machine M always rejects (soundness)

Construction of the RP machine:

Since L has a Las Vegas zero-knowledge proof system, there exists a probabilistic polynomial time machine M_V that simulates the membership proofs of P and V. Let $Q(|x|)$ denote an upper bound for the running time of M_V on input $x \in L$ (where Q is some polynomial). The Random Polynomial Time machine we build, M, will use M_V. On input x, machine M runs M_V on x, maintaining a step count. If M_V runs more than $Q(|x|)$ steps, or does not produce an accepting conversation, M rejects. Otherwise (if the conversation produced by M_V is accepting) M accepts.

Soundness of M:

Claim 4.1.1: On input $x \notin L$, machine M_V cannot possibly generate an accepting conversation.

Proof: Assume it could, that is there exists a random string r and a set of prover messages such that V running with random string r and receiving the appropriate messages accepts on x. Then the conversation could occur in a real interaction with non-zero probability, violating the conditions of Las Vegas protocols. □ Note that this claim follows only from the fact that accepting conversations cannot exist for $x \notin L$, and not from the fact that the conversation was generated by M_V. Therefore it is valid regardless of the "quality" of the texts produced by M_V. It is clear that M will never accept on $x \notin L$, and therefore the soundness condition is established.

Completeness of M: The completeness property of interactive proofs requires that conversations on $x \in L$ be accepting with very high probability. The same is clearly true of the conversations produced by M_V in the case of perfect zero-knowledge. Adapting the argument to computational zero-knowledge is simple in this case: note that ρ, the predicate used by V to decide whether to accept or reject, must be computable in polynomial time. Consequently, if M_V does not produce accepting conversations on $x \in L$ with very high probability, then ρ will distinguish the texts of the simulator from those of real interaction. □

We conclude that the error probability on $x \notin L$ instances, existing in all known zero-knowledge proofs, is inevitable and essential to the non-triviality of these proof systems. Another essential property of non-trivial zero-knowledge proofs is the randomness of the verifier. We prove this by demonstrating that any language which has a zero-knowledge interactive proof in which the verifier is deterministic, has a zero-knowledge Las Vegas interactive proof.

Lemma 4.1.1: Let P, V be a (zero-knowledge) interactive proof system for a language L, in which the verifier is deterministic. Then L has a (zero-knowledge) Las Vegas interactive proof.

Proof: We will show that if P, V is not itself Las Vegas, then either it can be slightly modified to become Las Vegas, or it cannot constitute an interactive proof system for L. Suppose the protocol is not Las Vegas. Then there exists a prover P^* and a set of $x \notin L$ such that V, when interacting with P^* on such an x accepts with non-zero probability. If this set is finite, then the protocol can be modified in the following way to become Las Vegas: on input x, the verifier first checks if x belongs to the "problematic" set, and if it does, V rejects immediately. Otherwise the original protocol is carried out. Clearly the modified protocol is Las Vegas. If the original protocol was zero-knowledge, so will be the modified protocol, since with respect to $x \in L$ both protocols are the same (recall that the definitions of zero-knowledge require nothing if $x \notin L$). We will now show that the "problematic" set must be finite: assume it is not, and there exists an infinite sequence Seq of $x \in L$ such that V, when interacting with P^* on $x \in Seq$ accepts with non-zero probability. Since V is deterministic, it follows that for every $x \in Seq$ there exists a sequence of prover messages that cause V to accept (that is V will accept with probability 1 when receiving this sequence of messages). Clearly there exists some $\hat{P}$ that for every $x \in Seq$ can find this sequence and always cause V to accept. One such $\hat{P}$ is a machine that given x simply tries out every possible set of messages to see on which of them, if any, V accepts. $\hat{P}$ can check this easily as the computation of V is completely determined by x and by the prover messages, and does not depend on some hidden random string. Therefore the protocol cannot be an interactive proof system for L. □ The following theorem is an immediate corollary of Theorem 4.1 and Lemma 4.1.1:

Theorem 4.2: Let L be any language and assume that L has a zero-knowledge interactive proof in which the verifier is deterministic. Then $L \in RP$.

4.3 One-Step Zero-Knowledge Proofs One-step interactive proof systems do exist and contain NP proof systems as a special case. However, NP-like proof systems give out a large amount of knowledge much of which is not essential for the proof. It was pointed out in [GMW1] that a one-step protocol cannot be zero- knowledge if it constitutes an interactive proof system for a language not in BPP. Here we present a formal proof of this statement. The proof holds even under the original [GMR1] definition of zero-knowledge.

Theorem 4.3: Let L be a language for which there exists a one-step zero-knowledge interactive proof system. Then $L \in BPP$.

Proof: As before, we will be using M_V, the simulator for the honest verifier V. The idea is to simulate the process of the interactive proof by ensuring that the message α generated by the simulator "on behalf" of the prover is *not* based on prior knowledge of the verifier's random string. V's decision on whether to accept or reject is obtained by evaluating a deterministic polynomial time predicate $\rho(x, \alpha, r)$, where x is the (common) input to P, V,

α is the prover's message to V and r is V's random string. If $x \in L$ then there exists some α such that for most r's the predicate must evaluate to ACC. In cases where $x \notin L$, for every α there may be a only few random strings r that cause ρ to evaluate to ACC, but the simulator may be such that on $x \notin L$ it always generates conversations in which ρ evaluates to ACC, using these few existing strings. (Recall that the definition of zero-knowledge requires nothing of the simulator in case $x \notin L$, and therefore this kind of behavior is possible). For that purpose we substitute the random string r produced by the simulator with a truly randomly chosen $r\prime$. In this way we simulate not the *text* but the *process* of the interactive proof, retaining its desired soundness property.

Construction of the BPP machine:

Following is a description of M, the BPP machine for L:

> On input x, machine M runs M_V on x, maintaining a step count. If M_V runs too long or does not produce an accepting conversation, M rejects. Otherwise, if $[x, r, \alpha]$ is an accepting conversation, where r is V's random string and α is the prover's message, M discards r, chooses a new, random string $r\prime$, and outputs $\rho(x, r\prime, \alpha)$.

Soundness of M: We claim that if $x \notin L$ and $r\prime$ is randomly chosen, then $\rho(x, r\prime, \alpha)$ will almost certainly evaluate to REJ, regardless of the value of α. Otherwise, if it evaluates to ACC with non-negligible probability for an infinite number of $x \notin L$, then the soundness condition of interactive proofs is violated.

Completeness of M: In the case of perfect zero-knowledge, the completeness of M follows directly from the completeness condition of interactive proofs. If $x \in L$ then the prover is guaranteed to produce (with high probability) an α that will cause V to accept for nearly all random strings r. The α's produced by the simulator will have the same property. The following lemma will adapt the proof to computational zero-knowledge.

Let $l_r(n)$ be the length of the random string used by V when interacting on input of length n.

Lemma 4.3.1: Let $\{P(x), V(x)\}_x$ and $\{M_V(x)\}_x$ be polynomially indistinguishable and let $\alpha(x)$ be the string output by M_V as the "prover message" when running on input x. Then for all but perhaps a finite set of $x \in L$ with very high probability $\rho(x, r, \alpha) = ACC$ when $x \in L$, if $r \in_R \{0,1\}^{l_r(|x|)}$.

Proof: In a manner similar to the proof of Theorem 4, we will use ρ to distinguish the text of simulation from those of real interaction. More formally: Assume there exists a constant $c0$ and an infinite sequence Seq of $x \in L$ for which the α produced by running $M_V(x)$, causes $\rho(x, r, \alpha)$ to evaluate to REJ with c-non-negligible probability, where $r \in_R \{0,1\}^{l_r(|x|)}$. Consider the following distinguisher, A: on input $H = [x, r, \alpha]$, the algorithm chooses $r\prime \in_R \{0,1\}^{l_r(|x|)}$ and computes $\rho(x, r\prime, \alpha)$. It then outputs 1 if the result is ACC and 0 otherwise.

If H is a description of a real conversation, then it follows from the completeness property of interactive proofs that A will output 1 with very high probability. We assumed that if H is a simulation text then A will output 0 with c-non-negligible probability. Therefore A will c-distinguish between $\{P(x), V(x)\}_x$ and $\{M_V(x)\}_x$, and the two distribution ensembles cannot be polynomially indistinguishable. $\Box$ The Theorem follows. $\Box$

4.4 Two-Step Auxiliary-Input Zero-Knowledge Proofs

We proceed to show that no two-step protocol can be auxiliary-input zero-knowledge in a non-trivial manner. Note that while one-step protocols cannot be (non-trivially) zero-knowledge even with respect to the prespecified verifier V, two-step protocols may be zero-knowledge (in a non-trivial manner) with respect to the prespecified verifier. In fact, such protocols (i.e., which *are* zero-knowledge with respect to V) are known for languages believed **not** to be in BPP (e.g., Quadratic Non-Residuosity [GMR1] and Graph Non-Isomorphism [GMW1]). Consequently, in order to prove our result we will have to make use of the full power of the definition of zero-knowledge, specifically the requirement that for *all* V^*'s there exists a simulator M_{V^*}. To prove an adapting lemma for this case we will need to assume a stronger definition of polynomial-indistinguishability, one in which the distinguishers are non-uniform (polynomial time machines). Let us present this definition:

Definition (non-uniform polynomial indistinguishability): For every algorithm A which has an auxiliary input tape, let $p_{A(z)}^{D(x,y)}$ denote the probability that A outputs 1 on input an element chosen according to the probability distribution $D(x, y)$ while having the string z as its auxiliary input. Denote by Dom the domain from which the pairs x, y are chosen. The distribution ensembles $\{D(x, y)\}_{x;y\in Dom}$ and $\{D\prime(x, y)\}_{x;y\in Dom}$ are *non-uniformly polynomially indistinguishable* if for every probabilistic algorithm (with auxiliary-input) A which runs in time polynomial in the length of its input, and for every constant $c0$, there exists N_0 such that for every x, $|x| > N_0$, for every y such that $(x, y) \in Dom$, and every z,

$$|p_{A(z)}^{D(x,y)} - p_{A(z)}^{D\prime(x,y)}| \leq |x|^{-c}.$$

We will refer to the definition of computational auxiliary-input zero-knowledge obtained when using the above definition of polynomial-indistinguishability as "non-uniform computational auxiliary-input zero-knowledge".

Remark 4.2: If we apply this definition of polynomial indistinguishability to blackbox-simulation zero-knowledge, the relationship demonstrated in section 3 still holds. Also, the proof of the Composition Theorem for the auxiliary-input definition (presented in section 3) can be carried out almost unaltered when using the above definition of polynomial indistinguishability.

We begin by an informal discussion: Two-step protocols can in general be viewed as ones in which the verifier generates questions which the prover can answer with non-negligible probability if and only if $x \in L$. When V follows the protocol, it "knows" the answer to its questions (and will therefore gain no knowledge from the answers), but this is no longer guaranteed for arbitrary V^*'s. The proof presented in this sub-section makes use of this observation to demonstrate the triviality of 2-step auxiliary-input protocols. It seems that the same reasoning should apply to the original [GMR1] definition. However, in view of the result of [AH2] discussed in the introduction (relativized 2-step [GMR1] zero-knowledge is not contained in relativized BPP), it is clear that the argument presented in this subsection will not extend to the [GMR1] definition, as it relativizes. In spite of that, it can be shown [O1] that the 2-step protocols mentioned above (for Quadratic Non-Residuosity and Graph Non-Isomorphism) cannot be [GMR1]-zero-knowledge unless these languages are in BPP. Both known two-step protocols mentioned above were modified by letting the verifier first "prove" to the prover that it "knows" the answers to its queries, resulting in protocols with more rounds which are zero-knowledge (with respect to any verifier)[GMR1, GMW1]. Returning to auxiliary-input zero-knowledge, we intend to prove:

Theorem 4.4: Let L be a language for which there exists a two-step perfect or non-uniformly computational auxiliary input zero-knowledge proof system. Then $L \in BPP$.

Proof: Let P, V be the 2-step proof system for L. Without loss of generality, we can describe P, V in the following way:

computes $\beta =$ *V(x,r)*

where r is V's random string

$V \rightarrow P$: β

computes $\alpha =$ *P(x,β)*

$P \rightarrow V$: α

computes $\rho(x, r, \alpha) \in \{ACC, REJ\}$ and stops.

The construction of the BPP machine in this case will run along the same general lines as in the one-step case, i.e. M will simulate the *process* of the interactive proof rather than merely its text. In a real interaction P must answer the "question" β without having access to the random string r used to compute β. The prover's ability to provide, under these conditions, an answer α for which $\rho(x, r, \alpha) = ACC$ is considered sufficient evidence that

$x \in L$. The completeness property of interactive proofs guarantees that the prover will be able to come up with such an α for almost any $\beta = V(x,r)$, if $x \in L$. The soundness condition of interactive proofs ensures that no prover could generate from $\beta = V(x,r)$ an α such that $\rho(x,r,\alpha) = ACC$ for any but a negligible fraction of the r's. Note that the prover is expected to generate such an α given only $\beta = V(x,r)$, wheras this α is tested againts r itself. As in our proof we intend to substitute the simulator for the prover as a means of generating α, it is essential that the random string r remain hidden from the simulator. Otherwise we could not rely on the soundness of the underlying interactive proof. Asking the simulator to "answer" our "question" β without giving away out secret r is achieved using the auxiliary input to the verifier.

Construction of the BPP machine: *Consider a verifier V^*, that given a string β^* as its auxiliary input sets $\beta = \beta^*$ (and sends β to P) instead of choosing a random r and computing $\beta = V(x,r)$.* Provided that the length of β^* is polynomial in the length of x, a verifier V^* as described above is clearly a polynomial time machine, for which a simulator M_{V^*} is guaranteed. Machine M_{V^*}, given as input $x \in L$ and any auxiliary input β^* simulates the interaction between P and V^*. Using M_{V^*} we now build M, the BPP machine for L. The idea is to generate a message β which is based on a truly random string r, and then to use M_{V^*} to obtain the prover message α corresponding to this β, without giving M_{V^*} access to r. The machine M will operate as follows:

On input x, machine M performs the following actions:

(1) Choose a random string r and compute $\beta^* = V(x,r)$.

(2) Run $M_{V^*}(x, \beta^*)$. If M_{V^*} produces a legal conversation $[x, \beta^*, r\prime, (\beta^*, \alpha)]$ ($r\prime$ is the random string generated by the simulator to emulate V^*'s random input in a real interaction), discard $r\prime$ and goto (3). Otherwise reject.

(3) Output $\rho(x, r, \alpha)$.

Soundness of M: Note that as far as V (or its simulated version) is concerned, we are exactly imitating the process of the interactive proof: a random string r is chosen and a message $\beta = V(x,r)$ computed. This message is sent to some other machine, which returns a message α. Then $\rho(x,r,\alpha)$ is used to determine whether to accept or reject. All we have done is substitute the simulator for the prover as a means of generating the message α. Therefore the soundness of M follows directly from the soundness condition of Interactive Proofs: if $x \notin L$ and M_{V^*} could generate an α for which $\rho(x,r,\alpha) = ACC$ with non-negligible probability, then a prover P^* using M_{V^*} could do the same, violating the soundness of the underlying interactive proof. It is clear therefore that M will reject any $x \notin L$ with very high probability.

Completeness of M: If $x \in L$ then P, when interacting with the prespecified V, is guaranteed to be able to generate an "answer" α such that $\rho(x,r,\alpha) = ACC$ for almost any random string r. Suppose now that P interacts with V^*, and that V^* has as auxiliary input a string β such that $\beta = V(x,r)$ for some randomly chosen r. Since r is randomly chosen and β is computed according to the protocol, a prover P has no way of knowing that it is interacting with a machine other than V, and will therefore behave exactly as when interacting with V, that is will attempt to generate an α such that $\rho(x,r,\alpha) = ACC$. The simulator in the case of perfect auxiliary-input zero-knowledge generates the same distribution as P, and will therefore also generate a suitable α. The completeness condition of interactive proofs can therefore be used here to establish the completeness of M. The following adapting lemma will show that this is true even for non-uniform computational zero-knowledge. Let $l_r(n)$ be the length of the random string used by V when interacting on input of length n.

Lemma 4.4.1: If $\{P(x), V^*(x,y)\}_{x;y}$ and $\{M_{V^*}(x,y)\}_{x;y}$ are non-uniformly polynomially indistinguishable then for all but perhaps a finite set of $x \in L$, if $\beta^* = V(x,r)$ for $r \in_R \{0,1\}^{l_r(|x|)}$ and α is obtained from the output of $M_{V^*}(x,\beta^*)$, then with very high probability $\rho(x,r,\alpha) = ACC$.

Proof: A history description H, originating either from $\{P(x), V^*(x,y)\}_{x;y}$ or from $\{M_{V^*}(x,y)\}_{x;y}$, will be of the form $H = [x, \beta^*, r\prime, (\beta^*, \alpha)]$, where β^* is the auxiliary input to V^* (used as the verifier's first message) and $r\prime$ is V^*'s random string. Observe that $r\prime$ almost certainly is not the random string r used to compute β^*, and is actually ignored by V^*. As stated earlier, the α generated by the prover is guaranteed by the completeness condition of Interactive Proofs to have the following property: α will cause $\rho(x,r,\alpha)$ to evaluate to ACC with very high probability, provided that r is the random string used to generate the β^*. If this property does not hold for the α's obtained from the output of $M_{V^*}(x,\beta^*)$, then a distinguisher testing for this property should be able to distinguish $\{P(x), V^*(x,y)\}_{x;y}$ from $\{M_{V^*}(x,y)\}_{x;y}$. However, given only H, the distinguisher has no idea which random string r was used to create β^* and therefore has no way to perform the required test. We will use the auxiliary input to the distinguisher, z, as a means to supply the distinguisher with the "true" random string corresponding to the conversation on its main input. Assume there exists a constant $c0$ and an infinite sequence Seq of $x \in L$ for which the α produced by running $M_{V^*}(x,\beta^*)$, where $\beta^* = V(x,r)$ and $r \in_R \{0,1\}^{l_r(|x|)}$, causes $\rho(x,r,\alpha)$ to evaluate to REJ with c-non-negligible probability. Denote by $p^M_{acc}(x,r)$ the probability that $\rho(x,r,\alpha)$ evaluates to ACC where $\beta^* = V(x,r)$ and α is obtained by running $M_{V^*}(x,\beta^*)$. Similarly $p^P_{acc}(x,r)$ will denote the probability that $\rho(x,r,\alpha)$ evaluates to ACC where $\beta^* = V(x,r)$ and α is obtained by running $P(x), V^*(x,\beta^*)$. Let $p^M_{acc}(x)$ be defined by

$$p^M_{acc}(x) = \sum_r \frac{1}{2^{l_r(|x|)}} \cdot p^M_{acc}(x,r)$$

and $p^P_{acc}(x)$ by

$$p^P_{acc}(x) = \sum_r \frac{1}{2^{l_r(|x|)}} \cdot p^P_{acc}(x,r)$$

By our assumption there exists some $c0$ such that for every $x \in Seq$

$$p^P_{acc}(x) - p^M_{acc}(x) \geq \frac{1}{|x|^c}$$

It follows that for every $x \in Seq$ there exists some r such that

$$p^P_{acc}(x, r) - p^M_{acc}(x, r) \geq \frac{1}{|x|^c}$$

Consider the following distinguisher A: on input a conversation $[x, \beta^*, r\prime, (\beta^*, \alpha)]$ and auxiliary input r, A computes $\rho(x, r, \alpha)$ and outputs 1 if the computation results in ACC. Clearly, for every $x \in Seq$ there exists some r and $\beta^* = V(x, r)$ such that A (running with auxiliary input r) will c-distinguish between $M_{V^*}(x, \beta^*)$ and $P(x), V^*(x, \beta^*)$. We conclude that $\{M_{V^*}(x, y)\}_{x;y}$ and $\{P(x), V^*(x, y)\}_{x;y}$ are not non-uniformly polynomially indistinguishable. □

The Theorem follows. □

4.5 Auxiliary-Input Zero-Knowledge Proof Systems With Deterministic Provers

In this subsection we show that any language which has an auxiliary-input zero-knowledge proof system in which the prover is deterministic belongs to BPP. The proof generalizes the proof method (but not the results) of the one-step and two-step cases. As in those cases, we intend to simulate the process of the interactive proof. Our proof relativizes, and thus in view of [AH2] will not extend to [GMR1] zero-knowledge.

Theorem 4.5: Let L be any language. If L has an auxiliary-input zero-knowledge proof system in which the prover is deterministic, then $L \in BPP$.

Proof: If P is deterministic, then the following holds: the entire conversation between P and V is fully determined by x and by r, the verifier's random string. Furthermore, P's i-th message α_i depends only on x and on $\beta_1, \cdots, \beta_i$. We will exploit this property in our proof. As in the one-step and two-step cases, we will imitate V's view of the interactive proof, using the simulator to generate the prover messages. We will begin by choosing a random string r, and construct the **unique** conversation corresponding to r and x round-by-round. At first, we will use the simulator to generate α_0 (and ignore the rest of the text). Once we have α_0, we can compute β_1 as V would, using the random string r. We will now run the simulator again, this time "forcing" the verifier to use the computed β_1 as its first message. This is achieved by placing β_1 on the verifier's auxiliary-input. Since the prover is deterministic (and the simulator must also be "deterministic in some sense" as we shall see) we can be sure that the same α_0 will be computed for the new conversation, and therefore the β_1 we computed will be a legal verifier message in the new conversation (that is, there exists a string r such that $\beta_1 = V(x, r, \alpha_0)$). From the new conversation we will obtain α_1, and so on. We thus reconstruct the entire conversation, while not revealing r to the simulator throughout the process. Once we have all the prover messages, we will use ρ to decide whether to accept or reject. It is easy to see that this method would not work if the prover were not deterministic. Consider for example a three-step protocol: we could first run the simulator to obtain (some) α_0. We could then compute a suitable β_1 and "force" the verifier to use it as its message. However, in the new conversation we would probably have a completely different α_0 (because P is not deterministic and may have more than one possible α_0) and the computed β_1 would no longer be a legal message in that conversation. As a result, we could not use the new conversation to obtain a meaningful α_1.

Construction of the BPP machine: consider a "verifier" V^* in the auxiliary input model, which when having a string $[\beta_1^*, \beta_2^*, ..., \beta_i^*]$ on its auxiliary input uses $\beta_1^*, \beta_2^*, ..., \beta_i^*$ as its i first messages to the prover, and then computes the rest of its messages in an arbitrary manner. Since the protocol is auxiliary-input zero-knowledge, there exists a probabilistic polynomial time machine M_{V^*} which simulates the interaction of V^* and P. We use M_{V^*} to build a BPP machine denoted M for the language L:

On input x, machine M proceeds as follows:

Choose random r.

Run $M_{V^*}(x)$ with empty auxiliary input (or simply $M_V(x)$, the simulator with respect to the prespecified verifier V) to obtain α_0 (discard the rest of the text).

For $i := 1$ to k do

Compute $\beta_i - V(x, r, \alpha_0, ..., \alpha_{i-1})$.

Run M_{V^*} with auxiliary input $[\beta_1, \beta_2, ..., \beta_i]$ to obtain α_i (which is our "guess" for $P(x, \beta_1, ..., \beta_i)$) . Discard the rest of the text.

enddo

output $\rho(x, r, \alpha_0, ..., \alpha_k)$

Soundness of M: As was the case for the 1-step and 2-step proofs, in this case we exactly imitate the process of the interactive proof as far as V is concerned, only substituting the simulator for the prover as a means of generating $\alpha_0, \cdots, \alpha_k$. The simulator computes α_i at stage i while having no knowledge of the random string used to compute $\beta_1, \cdots, \beta_i$, precisely the conditions under which P must compute α_i in a real interaction. It follows that M_{V^*}'s ability to generate, under these conditions, a set of messages $\alpha_0, \cdots, \alpha_k$ for which $\rho(x, r, \alpha_0, \cdots, \alpha_k)$ evaluates to ACC with non-negligible probability implies the ability of some prover P^* to do the same in a real interaction. The soundness of M therefore follows from the soundness of the underlying interactive proof. Note that the soundness condition does not depend in any way on the "zero-knowledge-ness" of the protocol.

Completeness of M: Consider an interaction on input x. Let $\beta_1, \cdots, \beta_i$ be the first i verifier messages of the unique conversation corresponding to x and to some random string r. The prover P, when interacting on x with a verifier that uses $\beta_1, \cdots, \beta_i$ as its first i messages, will output the messages $\alpha_0, \cdots, \alpha_i$ corresponding to x and r. This is true in particular for the previously described verifier V^*. Not until it receives the message β_{i+1} can P (perhaps) realize it is interacting with a cheater V^* and not with the well behaved V. Therefore all its messages up to that point will be as specified by the protocol. In the case of perfect zero-knowledge, the texts of M_{V^*} will have the same property. In particular, at round i the message α_i obtained from the simulation text will be the unique $P(x, \beta_1, \cdots, \beta_i)$ corresponding to x and the random string r chosen. In all, the set $\alpha_0, \cdots, \alpha_k$ of prover messages generated by M will be the unique set corresponding to x and r, and therefore the completeness of M follows from the completeness of P, V. We now proceed to adapt the argument to computational zero-knowledge. We need to prove that at round i, the messages $\alpha_0, \cdots, \alpha_i$ generated by M_{V^*} on input x and auxiliary input $[\beta_1, \cdots, \beta_i]$ are with very high probability $P(x), P(x, \beta_1), \cdots, P(x, \beta_1, \cdots, \beta_i)$. We will first address the following question:

Single Element Question:

Let $\{\pi_1^x\}_{x \in D}$ be a distribution ensemble having the following property: for every large enough x the probability distribution π_1^x assigns high probability to one element, denoted σ_x (in our case, the distribution created by the prover is totally deterministic, that is assigns probability 1 to some text σ_x). Let $\{\pi_2^x\}_{x \in D}$ be a distribution ensemble which is polynomially indistinguishable from $\{\pi_1^x\}_{x \in D}$. Must $\{\pi_2^x\}_x$ have essentially the same property (i.e. for every large enough x the distribution π_2^x assigns high probability to σ_x) ?

Before attempting to answer this question, let us examine more closely the notion of polynomial indistinguishability. In the definition of polynomial indistinguishability used throughout the paper, two distribution ensembles $\{\pi_1^x\}_x$ and $\{\pi_2^x\}_x$ claimed to be polynomially indistinguishable, must satisfy the following condition: any polynomial time probabilistic algorithm, on input a single string sampled from $\{\pi_1^x\}_x$, must behave approximately the same as when given a string sampled from $\{\pi_2^x\}_x$. Another, possibly stricter, definition is the following: any polynomial time algorithm, on input a *sequence* (of constant or polynomial size) of strings sampled from $\{\pi_1^x\}_x$ must behave approximately the same as when given a sequence of strings sampled from $\{\pi_2^x\}_x$. We will refer to the first version of the definition as *single-sample* polynomial indistinguishability, and to the second as *multiple-sample* polynomial indistinguishability. Multiple-sample polynomial indistinguishability bears relevance to our discussion due to the following fact: when using the multiple-sample definition one can easily prove a positive answer to the Single Element Question posed earlier, provided that $\{\pi_2^x\}_x$ can be sampled in polynomial time. The following two claims will demonstrate this.

Claim 4.5.1: Let $\{\pi_1^x\}_x$ assign high probability (say $\geq 3/4$) to σ_x for every large enough x and let $\{\pi_1^x\}_x$ and $\{\pi_2^x\}_x$ be multiple-sample polynomially indistinguishable. Then $\{\pi_2^x\}_x$ must for every large enough x assign very high probability (say 3/5) to exactly one string, denoted $\sigma_{x'}$.

Proof: Assume to the contrary that no string appears in π_2^x with high probability (i.e. higher than 3/5). Consider the following two-sample distinguisher A: on input two strings, s_1 and s_2, algorithm A outputs 1 if $s_1 = s_2$ and 0 otherwise. I f s_1, s_2 were sampled from $\{\pi_1^x\}_x$ then $s_1 = s_2$ with very high probability (namely, $\geq (3/4)^2$). On the other hand, if s_1, s_2 were sampled from $\{\pi_2^x\}_x$ then $s_1 = s_2$ with too low probability (namely, 13/25 9/16). Therefore A will distinguish $\{\pi_1^x\}_x$ from $\{\pi_2^x\}_x$. □ The above claim guarantees that $\{\pi_2^x\}_x$ assigns very high probability to a single string. We now show that this string must be σ_x (single-sample polynomial indistinguishability suffices to prove the following claim).

Claim 4.5.2: Let $\{\pi_1^x\}_{x\in D}$ be a distribution ensemble such that $\forall x \in D$ the distribution π_1^x assigns probability at least $\frac{1}{2} + \epsilon$ to one string, denoted σ_x. Let $\{\pi_2^x\}_{x\in D}$ be a distribution ensemble such that $\forall x \in D$ the distribution π_2^x assigns probability at least $\frac{1}{2} + \epsilon$ to one string, denoted $\sigma_{x'}$. If $\{\pi_1^x\}_x$ and $\{\pi_2^x\}_x$ are polynomially indistinguishable and $\{\pi_2^x\}_x$ can be sampled in polynomial time, then for all but finitely many $x \in D$ the string σ_x equals the string $\sigma_{x'}$.

Proof: If otherwise, consider the following distinguisher A: on input a string s, algorithm A samples π_2^x to obtain, with overwhelmingly high probability, the string $\sigma_{x'}$ (which by hypothesis is different from σ_x). (The number of sample points is polynomial in $1/\epsilon$.) The algorithm outputs 1 if $s = \sigma_{x'}$ and 0 otherwise. If s comes from π_2^x, then with probability $\geq \frac{1}{2} + \epsilon$ we have $s = \sigma_{x'}$. If, on the other hand, s comes from π_1^x then with probability $\geq \frac{1}{2} + \epsilon$ we have $s = \sigma_x$ and hence (assuming $\sigma_{x'} \neq \sigma_x$) $s \neq \sigma_{x'}$ (with probability $\geq \frac{1}{2} + \epsilon$). Therefore A will distinguish $\{\pi_1^x\}_x$ from $\{\pi_2^x\}_x$. □ All that remains in order to answer the Single Element Question for single-sample polynomial indistinguishability is to show, if we can, that single-sample polynomial indistinguishability is equivalent to multiple-sample polynomial indistinguishability. But can we? Polynomial indistinguishability was originally discussed in the context of probabilistic encryption [GM] and pseudorandom generators [Y]. In these cases the distributions of *both* the ensembles which are assumed to be polynomially indistinguishable can be sampled in polynomial time. This fact can be used to prove that in these contexts single-sample polynomial indistinguishability (the usual definition) and multiple-sample polynomial indistinguishability are equivalent (intuitively, because the distinguisher can generate additional samples by itself).* The same proof cannot be applied, however, in general and in particular in the context of zero-knowledge, because in this case one of the distribution ensembles, mainly P, V, is *not* polynomial time samplable. [1] We return to our original question. Since we cannot demonstrate the equivalence single-

[1]* In [GGM] Goldreich, Goldwasser and Micali define multiple-sample polynomial indistinguishability and prove its equivalence to single-sample polynomial indistinguishability in the context of pseudorandom generators.

sample polynomial indistinguishability to multiple-sample polynomial indistinguishability, we must adopt a different approach. We now demonstrate that even under single-sample polynomial-indistinguishability, $\{\pi_2^x\}_x$ must assign very high probability to σ_x (the string assigned high probability by π_1^x). For any distribution π and string s, we denote by $\pi(s)$ the probability assigned by π to s.

Single Element Lemma: Let $\epsilon \leq 1/5$. Let $\{\pi_1^x\}_{x \in D}$ and $\{\pi_2^x\}_{x \in D}$ be polynomially indistinguishable distribution ensembles such that π_1^x and π_2^x are probability distributions over strings of length polynomial in $|x|$. Assume that for every large enough x there exists some string, denoted σ_x, such that π_1^x assigns to σ_x probability $\geq 1 - \epsilon$. Assume further that $\{\pi_2^x\}_{x \in D}$ is polynomial-time samplable (though $\{\pi_1^x\}_{x \in D}$ may not be). Then $\pi_2^x(\sigma_x) 1 - 2\epsilon$, for all but finitely many x.

Proof: Assume there exist an infinite sequence *Seq* of x's such that π_2^x assigns σ_x probability at most $1 - 2\epsilon$. For any $x \in Seq$ there must be one or more strings s such that $\pi_2^x(s)\ 0$ (σ_x may or may not be one of them). These strings can be arranged in lexicographical order. For any two strings s_1, s_2, we will write $s_1\ s_2$ to mean that s_1 precedes s_2 in lexicographical order. $s_1 \leq s_2$ will mean $s_1\ s_2$ or $s_1 = s_2$. Let P_x^- be defined by

$$P_x^- = \sum_{\{s \mid s\ \sigma_x\}} \pi_2^x(s)$$

and P_x^+ by

$$P_x^+ = \sum_{\{s \mid s \leq \sigma_x\}} \pi_2^x(s)$$

> **Example:** Let $\sigma_x = 100$ and π_2^x assign probability $1/5$ to each of the following strings: 00, 01, 000, 100, 1001. Then $P_x^- = 3/5$ and $P_x^+ = 4/5$. If ($\sigma_x = 100$ and) π_2^x assigns probability $1/4$ to each of the strings 00, 01, 000, 1001, then $P_x^- = P_x^+ = 3/4$.

For any $x \in Seq$, we have three possible cases (not necessarily distinct):

(1) $P_x^+\ 0.8$

(2) $P_x^-\ 0.2$

(3) $P_x^+ \geq 0.8$ and $P_x^- \leq 0.2$

Denote by S_i, $1 \leq i \leq 3$, the subsequence of *Seq* such that $x \in S_i$ if case i holds for x. Clearly, at least one of the subsequences must be infinite. We will now show how to handle each of the corresponding cases.

Case (1): Assume S_1 is infinite. Let us first prove the following claim:

Claim 4.5.3: Let π be any probability distribution on strings. A $k-experiment$ on π will consist of sampling k times the distribution π. Denote by s_i, $i \leq k$, the result of the i-th sampling in the k-experiment. Let P_i denote the probability that the sample s_i is larger or equal to all of the samples (i.e. $P_i = Prob(\forall j s_i \geq s_j)$). Then $P_1 \geq \frac{1}{k}$.

Proof: For reasons of symmetry, $\forall i, j \leq k P_i = P_j$. Since in every k-experiment there must be at least one maximal value, it follows that $\sum_{i=1}^{k} P_i \geq 1$, and therefore $\forall i \leq k, P_i \geq \frac{1}{k}$. □
Consider now the following distinguisher, A: on input a string s, the distinguisher A first samples π_2^x for $k-1$ times (k is a constant to be determined latter). It then outputs 1 if s is greater than or equal to each of the $k-1$ sampled strings. Suppose s was sampled from π_2^x. We can view the whole process as a k-experiment on π_2^x, in which s is the first sample. By the above claim, the probability that A outputs 1 in this case is greater than or equal to $\frac{1}{k}$. On the other hand, if s was sampled from π_1^x (in which case $s = \sigma_x$ with probability $\geq 1 - \epsilon$), then (for every $x \in S_1$) the probability of a single sample being smaller than or equal to s is less than $(1-\epsilon) \cdot 0.8 + \epsilon 0.9$ (the first term is for the case $s = \sigma_x$). The probability that all $k-1$ samples will be smaller than or equal to s, is thus less than 0.9^{k-1}. A suitable choice of k (say $k = 50$) yields 0.9^{k-1} $1/(2k)$. Clearly, for any $x \in S_1$, algorithm A will distinguish between π_1^x and π_2^x, and therefore the distribution ensembles cannot be polynomially indistinguishable.

Case (2): Assume S_2 is infinite. This case is symmetric to the previous case, since if we reverse the lexicographical order we obtain P_x^+ 0.8. It can therefore be handled in the same way.

Case (3): Assume S_3 is infinite. In this case reversing the lexicographical order will still leave us in the same case. Observe however, that if $P_x^+ \geq 0.8$ and $P_x^- \leq 0.2$, it must be that $\pi_2^x(\sigma_x) \geq 0.6$. In such a case we can find σ_x with sufficient confidence by sampling π_2^x a polynomial number of times. Consider a distinguisher A which on input a string s samples π_2^x enough times to pick out with very high probability (say $1-\epsilon/3$) a string s' for which $\pi_2^x(s') \geq 0.6$, and then outputs 1 if $s = s'$. We have $Prob(s' = \sigma_x) \geq 1 - \epsilon/3$. For any x, $x \in S_3$, if s was sampled from π_1^x then $Prob_{\pi_1^x}(s = s') \geq 1 - \epsilon - \epsilon/3$. We started out the proof by assuming that for any $x \in Seq$ (and therefore for any $x \in S_3$) $\pi_2^x(\sigma_x) \leq 1 - 2\epsilon$ and hence $Prob_{\pi_2^x}(s = s') \leq 1 - 2\epsilon + \epsilon/3$. Clearly, A will distinguish between $\{\pi_1^x\}_{x \in D}$ and $\{\pi_2^x\}_{x \in D}$ (with gap $\epsilon/3$). The lemma follows. □

The adapting lemma we need to prove is an easy consequence of the Single Element Lemma.

Lemma 4.5.1: Let D be the set of all pairs (x, y) for which $x \in L$ and $y = [\beta_1, \cdots, \beta_i]$, such that for some random string r

$$\beta_1 = V(x, r, [\alpha_0 = P(x)])$$
$$\beta_2 = V(x, r, [\alpha_0 = P(x), \alpha_1 = P(x, \beta_1)])$$
$$\cdots$$

$$\beta_i = V(x, r, [\alpha_0 = P(x), \alpha_1 = P(x, \beta_1), \cdots, \alpha_{i-1} = P(x, \beta_1, \beta_2, \cdots, \beta_{i-1})])$$

If the distribution ensembles $\{M_{V^*}(x,y)\}_{(x,y)\in D}$ and $\{P(x), V^*(x,y)\}_{(x,y)\in D}$ are polynomially indistinguishable then in the texts produced by M_{V^*} on input x and y with very high probability

$$\forall j \leq i \alpha_j = P(x, \beta_1, \beta_2, \cdots, \beta_j)$$

Proof: Let $\pi_1^{(x,y)}$ be the distribution of the first $i+1$ prover messages in P's interaction with V^* on input x and auxiliary input $y = [\beta_1, \cdots, \beta_i]$ (note that this distribution depends solely on x and y and not on V^*'s random string). Let $\pi_2^{(x,y)}$ be the distribution of the first $i+1$ "prover messages" produced by M_{V^*} on input x and y. Clearly, if $\{P(x), V^*(x,y)\}_{(x,y)\in D}$ and $\{M_{V^*}(x,y)\}_{(x,y)\in D}$ are polynomially indistinguishable, then so are $\{\pi_1^{(x,y)}\}_{(x,y)\in D}$ and $\{\pi_2^{(x,y)}\}_{(x,y)\in D}$. The ensemble $\{\pi_2^{(x,y)}\}_{(x,y)\in D}$ clearly is polynomial-time samplable, and $\pi_1^{(x,y)}$ assigns one string (namely $[P(x), P(x,\beta_1), \cdots, P(x, \beta_1, \cdots, \beta_i)]$) probability 1 for any $(x,y) \in D$. We can thus apply the Determinicity Lemma. We conclude that at round i, machine M_{V^*} will produce with very high probability the messages $P(x), \cdots, P(x, \beta_1, \cdots, \beta_i)$ corresponding to x and the string r used to compute $\beta_1, \cdots, \beta_i$. $\square$ The Theorem follows. $\square$

Remark 4.3 The fact that single-sample and multiple sample polynomial indistinguishability may not be equivalent in the context of zero-knowledge raises the following questions, which deserve further investigation: can single-sample and multiple-sample polynomial indistinguishability be proved equivalent or strictly different in the context of zero-knowledge (i.e. when one ensemble is not polynomial-time samplable) ? And if they are different, which should be used in a "correct" cryptographic definition of zero-knowledge ? Observe that the two definitions are equivalent if the distinguisher is allowed to have auxiliary input, as in the definition of "non-uniform" polynomial indistinguishability presented in the previous sub-section.

4.6 A Remark on Extension to Zero-Knowledge Arguments

The results of the previous subsections extend to the zero-knowledge arguments introduced in [BCC]. In these protocols it is guaranteed that there exists no *efficient* way of fooling the verifier to accept false statements. This is a relaxion of the soundness condition in interactive proofs where it is required that there exists no way of fooling the verifier (to accept false statements). In the extensions we use exactly the same constructions of BPP machines, and the same reasoning for the completeness condition (i.e. that the machine accepts, with high probability, inputs in the language). For the soundness of the BPP machine (i.e. showing that it rejects, with high probability, inputs not in the language) we use a slightly more careful reasoning. Recall that the soundness of the BPP machine is proved by relying on the soundness of the protocol. In fact, in all cases we have shown that a violation of the soundness of the BPP machine yields violation of the soundness condition for interactive proofs. This, in turn, was done by incorporating the "cheating BPP machine" inside of a "cheating prover". Hence, the "cheating prover" constructed in all cases is indeed efficient and thus contradicts the soundness condition of zero-knowledge arguments as well.

ACKNOWLEDGEMENTS We would like to thank Shimon Even for making useful comments on the paper. The concept of Las Vegas interactive proofs was raised by Manuel Blum and communicated through Silvio Micali. The question of triviality of proof systems with deterministic provers was raised by Shimon Even.

REFERENCES

[B] Babai, L., "Trading Group Theory for Randomness", *Proc. 17th STOC*, 1985, pp. 421-429.

[BCC] Brassard, G., D. Chaum, and C. Crepeau, "Minimum Disclosure Proofs of knowledge", *JCSS*, Vol. 37, No. 2, Oct. 1988, pp. 156-189.

[F] Fortnow, L., "The Complexity of Perfect Zero-Knowledge", *Proc. of 19th STOC*, 1987, pp. 204-209.

[FS] Feige, U., and A. Shamir, personal communication.

[GGM] Goldreich, O., S. Goldwasser, and S. Micali, "How to Construct Random Functions", *Jour. of ACM*, Vol. 33, No. 4, 1986, pp. 792-807.

[GK] Goldreich, O., and H. Krawczyk, "On the Composition of Zero-Knowledge Proof Systems", to appear in the proceedings of *17th ICALP*, 1990.

[GMS] Goldreich, O., Y. Mansour, and M. Sipser, "Interactive Proof Systems: Provers that Never Fail and Random Selection", *Proc 28th FOCS*, 1987.

[GMW1] Goldreich, O., S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design", *Proc. 27th FOCS*, 1986, pp. 174-187.

[GMW2] Goldreich, O., S. Micali, and A. Wigderson, "How to Play Any Mental Game or A Completeness Theorem for Protocols with Honest Majority", *Proc. of 19th STOC*, 1987, pp. 218-229.

[GMR1] Goldwasser, S., S. Micali, and C. Rackoff, "Knowledge Complexity of Interactive Proofs", *Proc. 17th STOC*, 1985, pp. 291-304.

[GMR2] Goldwasser, S., S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", *SIAM J. on Comput.*, Vol. 18, No. 1, 1989, pp. 186-208.

[GM] Goldwasser, S., and S. Micali, "Probabilistic Encryption", *JCSS*, Vol. 28, No. 2, 1984, pp. 270-299.

[GS] Goldwasser, S., and M. Sipser, "Arthur Merlin Games versus Interactive Proof Systems", *Proc. 18th STOC*, 1986, pp. 59-68.

[O1] Oren,Y., "Properties of Zero-Knowledge Proofs", M.Sc. Thesis, Computer Science Dept., Technion, Israel, Nov. 1987 (in Hebrew).

[O2] Oren, Y., "On the Cunning Power of Cheating Verifiers: Some Observations About Zero-Knowledge Proofs", *Proc. 28th FOCS*, 1987, pp. 462-471.

[TW] Tompa, M., and H. Woll, "Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information", *Proc. 28th FOCS*, 1987, pp. 472-482.

[Y] Yao, A.C., "Theory and Applications of Trapdoor Functions", *Proc. 23rd FOCS*, 1982, pp. 80-91.

[AH1] Aiello, W., and J. Hastad, "Perfect Zero-Knowledge Languages Can Be Recognized in Two Rounds", *28th FOCS*, 1987, pp. 439-448.

[AH2] Aiello, W., and J. Hastad, "Relativized Perfect Zero-Knowledge is not BPP", *Information and Computation*, Vol. 93, 1992, pp. 223-240.

[IY] Impagliazzo, R. and Yung, M., "Direct Minimum-Knowledge Computations", *Advances in Cryptology – Crypto87* (proceedings), Lecture Notes in Computer Science, Vol. 293, Springer-Verlag, New-York, 1987, pp. 40-51.

[S] A. Shamir, IP=PSPACE, *31st FOCS*, 1990, pp. 11-15.