ON THE CUNNING POWER OF CHEATING VERIFIERS: SOME OBSERVATIONS ABOUT ZERO KNOWLEDGE PROOFS

(Extended abstract)

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 various definitions of zero-knowledge. Two definitions which are of special interest are *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 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 *R*. 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.

In order to derive most of the results in the paper we make use of the full power of the definition of zero-knowledge: specifically, the requirement that there exist a simulator for *any* verifier, including "cheating verifiers".

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 he can compute by himself 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 properties of these proof systems.

Research was partially supported by the Fund for Basic Research Administered by the Israeli Academy of Sciences and Humanities.

The original definition of zero-knowledge was presented in [GMR1]. Although argued too restrictive by some, it seems to be unsatisfactory when considering cryptographic settings. 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 his messages based on information acquired *before* the proof protocol begun, 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 order to consider this kind of "cheating", 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, he can also compute by himself when having access to the same information. 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*. The fact that such a strong result can be derived indicates that the auxiliary-input zero-knowledge definition is suitable for cryptographic purposes.

The importance of the definition of *auxiliary−input* zero-knowledge is examplified by the major role of the auxiliary-inputs in the proof of the *composition* theorem. The composition theorem is a generic special case of the use of zero-knowledge proofs within cryptographic protocols. Intuitively, it states that the sequential application of zero-knowledge proofs yields a zero-knowledge proof. 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. (It seems that the original definition is not closed under composition *.)

The requirements of the auxiliary-input definition may seem very restrictive. However, all known zero-knowledge proof systems [GMR1, GMW1] in fact satisfy 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].

***) This observation, as weel as its resolution using "non-uniform" verifiers, was made independently by Goldwasser, Micali and Rackoff [GMR2], Tompa and Woll [TW] and Feige and Shamir [FS].**

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. 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 $x \notin L$ 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* [GMS].

It is easy to see that the class of languages membership in which can be proved by a deterministic prover equals that for which membership can be proved by a probabilistic prover. 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 $x \notin L$ instances and randomness of the prover 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.

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-Residuocity [GMR1] and Graph Non-Isomorphism [GMW1]). Therefore, unlike Fortnow [F] who actually relies 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".

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 various 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 various triviality results.

2. NOTATION AND BASIC DEFINITIONS

We recall the definition of *interactive proof systems** [GMR1]: 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 his predetermined program, V, the following two conditions hold:

1) *Completeness of the interactive proof system:* If the common input x is in L and the prover runs its predetermined program, P, then the verifier accepts x with probability at least $1-|x|^{-c}$, for every constant $c>0$ and large enough x. In other words, the prover can convince the verifier of $x \in L$.

2) *Soundness of the interactive proof system:* If the common input x is NOT in L, then for every program P^*, run by the prover, the verifier rejects x with probability at least $1-|x|^{-c}$ (for every constant $c>0$ and large enough x). 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 probability distributions indexed by two parameters, which are treated differently, and for non-uniform distinguishers.

Definition (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 $D(x,y)$ and 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 $\{D'(x,y)\}_{x;y \in Dom}$ are *polynomially indistinguishable* if for every probabilistic algorithm (with auxiliary-input) A which runs time polynomial in the length of its input, for every constant $c>0$ there exists N_0 such that for every x, $|x| > N_0$, and for every y such that $(x,y) \in Dom$, and every z,

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

Note that we do not require $|y|>N_0$. The original definition is obtained from the above definition by omitting all mention of y and z. We will occasion-

* An alternative definition due to Babai [B] was shown equivalent by Goldwasser and Sipser [GS].

ally avoid specifying the domain, and write $\{D(x,y)\}_{x;y}$ instead of $\{D(x,y)\}_{x;y \in Dom}$.

3. A TAXONOMY OF ZERO-KNOWLEDGE DEFINITIONS

In this section we present several alternative definitions of the notion of zero-knowledge, and investigate the relationships 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 input 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 denote by $<P(x),V^*(x)>$ 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 $\{M_{V^*}(x)\}_x$ and $\{<P(x),V^*(x)>\}_x$ are polynomially indistinguishable.

Remark: 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' whose output is the *history* of the interaction of V^* with P, it follows that 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* 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 he is computing his 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 him. In order to do so the adversary may deviate from the specified program and compute his messages in a manner suited to his goals. Most probably he will want to base the computation of his 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 will enable the "verifier" to compute the desired messages will be entered.

Definition (Auxiliary-input): 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 for every $x \in L$ and every auxiliary input y 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 $D_1 = \{(x,y) \mid x \in L\}$.

Providing the distinguisher with the auxiliary input z is not as strongly motivated. It is required however, for the triviality proofs of 2-step and deterministic-prover protocols, discussed in section 4. All other results in the paper do not depend on this issue.

The second definition we consider, which will be referred to as *Universal Simulation*, is not as natural as the other two, but will help us in proving our results in subsection 3.3. This definition requires that for every polynomial Q there exist a single universal probabilistic polynomial time machine M_Q that given the code of any V^* whose running time is bounded by Q, simulates the interaction of V^* with the prover P on any $x \in L$. Denote by $Time_P^{V^*}$ a function bounding the running time of V^* when interacting with the prover P (namely, on input x, machine V^* makes at most $Time_P^{V^*}(|x|)$ steps). Denote by $EN(V^*)$ the encoding of a machine V^*.

Definition (Universal simulation): An interactive proof system for a language L is *universal simulation zero-knowledge* if for every polynomial Q there exists a probabilistic polynomial time machine M_Q such that for every probabilistic machine V^* for which $Time_P^{V^*} = O(Q)$ the distribution ensembles $\{<P(x),V^*(x)>\}_{x;V^* \in D_2}$ and $\{M_Q(EN(V^*),x)\}_{x;V^* \in D_2}$ are polynomially indistinguishable, where $D_2 = \{(x,V^*) | x \in L \wedge Time_P(V^*) = O(Q)\}$.

A variation of this definition allows the universal simulator only to use V^* as a blackbox (in the sense defined below), while having no access to its code. The running time of V^* will not be counted in the running time of simulation, and therefore we will require the existence of a *single* simulator M_u. A suitable name for this definition would be *Blackbox Simulation* zero-knowledge.

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): 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 probabilistic polynomial time machine V^* and for every $x \in L$, the distribution ensembles $\{<P(x),V^*(x)>\}_{x;V^* \in D_3}$ and $\{M_u^{V^*}(x)\}_{x;V^* \in D_3}$ are polynomially indistinguishable, where $D_3 = \{(x,V^*) | x \in L\}$.

3.2 Proof of the Composition Theorem

We first define the notion of a 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 *composition* of the k protocols, $<P,V>$ 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.

Theorem 1: 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 that $<P,V>$ is auxiliary-input zero-knowledge. The following proof can be carried out without using auxiliary input to the distinguisher (just omit all mention of z). That is, the composition theorem holds also in the case where the distinguishers are uniform (both in the Theorem's hypothesis and in its conclusion). 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 small-print paragraphs throughout the proof is to provide insight and intuition to the otherwise rather formal proof.

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.

In the proof we will construct an M_{V^*} for every V^*. 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 $M_{V^*}^1$, $M_{V^*}^2$, ...,$M_{V^*}^k$, which simulate the interaction of V^* with $P_1, P_2, \ldots, P_k$, 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 $M_{V^*}^{i+1}$ should be its final state in the simulation by $M_{V^*}^i$. This can be achieved by placing information which will enable V^* to reconstruct the final state of the i-th stage on its auxiliary-input for the $i+1$-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' which will have the required behavior.

As a first step we will consider a verifier V' 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, V' 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 $m=\varepsilon$ (the empty string) and y is not itself a history description then V' only copies x, y, r to the work tape of its built-in version of V^* and then "behaves" like V^*. Machine V' actually always ignores its "real" random string. In all other senses V' 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 $<P(x),V'(x,[x,y,r,\varepsilon])>$.

Construction of the simulator for V^*:

Since the individual protocols are assumed to be auxiliary-input zero-knowledge, there exists machines $M_{V'}^1$, $M_{V'}^2$, ... , $M_{V'}^k$ which simulate the history of V''s interaction with P_1, P_2, ...,P_k, respectively. The output produced by $M_{V'}^i$ will be a string h_i of the form $[x_i,h_{i-1},r_i,m_i]$, where r_i is V''s random string (which is actually ignored) in this simulation and m_i is the sequence of messages sent "on behalf" of the prover. Let s_1s_2 denote the concatenation of strings s_1 and s_2. We now describe M_{V^*}. On input x and y, machine M_{V^*} runs

```
    choose random string r
    $h_0 \leftarrow [x,y,r,\varepsilon]$
    $h_1 \leftarrow M_{V'}^1(x_1,h_0)$
    $h_2 \leftarrow M_{V'}^2(x_2,h_1)$
    ...
    $h_k \leftarrow M_{V'}^k(x_k,h_{k-1})$
    $m \leftarrow m_1 m_2 \cdots m_k$
    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: There cannot exist a test A which for infinitely many triplets (x,y,z), where z is the auxiliary input to A, distinguishes between $M_{V^*}(x,y)$ and $<P(x),V^*(x,y)>$.

Proof: Suppose there exists such a test A.

We will show that in such a case there exists another test $A^{(i)}$ that for some i and for infinitely many triplets (x_i,y_i,z) distinguishes between $M_{V'}^i(x_i,y_i)$ and $<P_i(x_i),V'(x_i,y_i)>$, contrary to the assumption that $M_{V'}^i$ 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
    $h_0 \leftarrow [x,y,r,\varepsilon]$
    $h_1 \leftarrow <P_1(x_1),V'(x_1,h_0)>$
    $h_2 \leftarrow <P_2(x_2),V'(x_2,h_1)>$
    ...
    $h_i \leftarrow <P_i(x_i),V'(x_i,h_{i-1})>$
    $h_{i+1} \leftarrow M_{V'}^{i+1}(x_{i+1},h_i)$
    ...
```

```
$h_k \leftarrow M_{V'}^{k}(x_k, h_{k-1})$,
$m \leftarrow m_1 m_2 \cdots m_k$
$OUTPUT([x, y, r, m])$.
```

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 distinguish between the extreme hybrids, then there must exist two adjacent hybrids which can also be distinguished, say H_i and H_{i+1}.

Let $pref_i(x,y)$ be the probability distribution defined by the process

```
choose a random string $r$
$h_0 \leftarrow [x, y, r, \varepsilon]$
$h_1 \leftarrow <P_1(x_1), V'(x_1, h_0)>$
$h_2 \leftarrow <P_2(x_2), V'(x_2, h_1)>$
...
$h_i \leftarrow <P_i(x_i), V'(x_i, h_{i-1})>$
$OUTPUT(h_i)$.
```

Let h be a string which may occur with non-zero probability in either of the distributions $M_{V'}^{i}(x_i, h_{i-1})$, and $<P_i(x_i), V'(x_i, h_{i-1})>$, where h_{i-1} is a string assigned non-zero probability by $pref_{i-1}(x,y)$. 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

```
$h_{i+1} \leftarrow M_{V'}^{i+1}(x_{i+1}, h)$
$h_{i+2} \leftarrow M_{V'}^{i+2}(x_{i+2}, h_{i+1})$
...
$h_k \leftarrow M_{V'}^{k}(x_k, h_{k-1})$
$m \leftarrow m_1 m_2 \cdots m_k$
$OUTPUT([x, y, r, m])$.
```

The distribution $pref_i(x,y)$ is actually a distribution on all the possible auxiliary inputs to the $i+1$-st stage, given that the initial input is x and the initial auxiliary input is the string y. $suff_i$ 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 $M_{V'}^{i}(x_i, h_{i-1})$ then the effect of $suff_i$ will correspond to a string coming from $H_i(x,y)$. If the input comes from $<P(x_i), V'(x_i, h_{i-1})>$, then the effect of $suff_i$ will correspond to a string coming from $H_{i+1}(x,y)$. Our aim is to show that if (x,y) are such that A 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 distinguish between between $M_{V'}^{i}(x_i, h^*)$ and $<P_i(x_i), V'(x_i, h^*)>$. $A^{(i)}$ 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:

$PR_{i-1}[h] = Prob\{pref_{i-1}(x,y) = h\}$
$suff_i(M[h]) = suff_i(M_{V'}^{i}(x_i, h))$
$suff_i(P[h]) = suff_i(<P_i(x_i), V'(x_i, h)>)$

The following relationships hold:

$$p_{A(z)}^{H_i(x,y)} = \sum_h PR_{i-1}[h] \cdot p_{A(z)}^{suff_i(M[h])}$$

Recall that $p_{A(z)}^{D}$ denotes the probability that algorithm A outputs 1 on input an element chosen according to the probability distribution D, and while having z as its auxiliary input.

The probability $p_{A(z)}^{H_i(x,y)}$ can be written 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 $i-1$-st stage history.

Similarly:

$$p_{A(z)}^{H_{i+1}(x,y)} = \sum_h PR_{i-1}[h] \cdot p_{A(z)}^{suff_i(P[h])}$$

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

Since $p_{A(z)}^{H_i(x,y)} - p_{A(z)}^{H_{i+1}(x,y)} > \frac{1}{k \cdot |x|^c}$ there exists some h^* for which

$$p_{A(z)}^{suff_i(M[h^*])} - p_{A(z)}^{suff_i(P[h^*])} > \frac{1}{k \cdot |x|^c}$$

We conclude that for every (x,y,z) for which $H_0(x,y)$ and $H_k(x,y)$ can be distinguished there exists (x_i, y_i, z) such that $<P_i(x_i), V'(x_i, y_i)>$ and $M_{V'}^{i}(x_i, y_i)$ can be 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 $M_{V'}^{i}(x_i, y_i)$, and auxiliary input z, 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

```
$h_{i+1} \leftarrow M_{V'}^{i+1}(x_{i+1}, T)$
$h_{i+2} \leftarrow M_{V'}^{i+2}(x_{i+2}, h_{i+1})$
```

```
...
$h_k \leftarrow M_{V'}^{k}(x_k, h_{k-1})$
$m \leftarrow m_1 m_2 \cdots m_k$
$OUTPUT([x, y, r, m])$
```

to obtain a text $T'=[x,y,r,m]$. The test $A^{(i)}$ then runs A on T' with auxiliary input z and outputs the output of A.

By our construction it is clear that $A^{(i)}(z)$ will distinguish between $<P_i(x_i), V'(x_i, y_i)>$ and $M_{V'}^{i}(x_i, y_i)$. This contradicts the fact the $M_{V'}^{i}$ is a "good" simulator for $<P_i, V'>$. □

We conclude that $\{M_{V'}(x,y)\}_{x;y}$ is polynomially indistinguishable from $\{<P(x), V^*(x,y)>\}_{x;y}$ and the Theorem follows. □

3.3 Relationships Between the Definitions

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

Theorem 2:

(1) $Cl(auxiliary-input) \subseteq Cl([GMR1])$

(2) $Cl(blackbox-simulation) \subseteq Cl(universal-simulation)$

(3) $Cl(universal-simulation) \subseteq Cl([GMR1])$

For example, to see why statement (1) is true, observe that a verifier in the [GMR1] model can be considered as a verifier with auxiliary input which it always empty, and thus [GMR1] zero-knowledge is a special case of auxiliary-input zero-knowledge.

Next, we establish the equivalence of the auxiliary-input and the universal-simulation models:

Theorem 3:

$Cl(auxiliary-input) = Cl(universal-simulation)$.

Proof is omitted.

The following is an immediate result of Theorems 2 and 3:

Corollary:

$Cl(blackbox-simulation) \subseteq Cl(auxiliary-input)$

Remark: The relationships derived in the above theorems hold also for *perfect zero-knowledge* and *almost-perfect zero-knowledge*. (see [F] for definitions).

4. THE TRIVIALITY OF CERTAIN CLASSES OF ZERO-KNOWLEDGE PROOF SYSTEMS

In this section we demonstrate some essential properties of zero-knowledge proof systems. We do so by showing that only languages in BPP can have proof systems that do not possess these properties.

Remark: In this extended abstract most proofs are omitted.

4.1 1-Step Zero-Knowledge Proof Systems

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. In the full version of this paper we present a formal proof of this statement. The proof holds under the original [GMR1] definition of zero-knowledge and thus also under all other definitions mentioned in this paper.

Theorem 4: Let L be a language for which there exists a one-step zero-knowledge interactive proof system. Then $L \in BPP$.

Using a similar technique we can show:

Theorem 5: Let L be a language for which there exists a zero-knowledge interactive proof system in which the verifier is deterministic. Then $L \in BPP$.

4.2 2-Step Auxiliary-Input Zero-Knowledge Proof Systems

We proceed to show that no two-step protocol can be auxiliary-input-zero-knowledge in a non trivial manner. (By theorem 3 and its corollary this holds also for the universal and blackbox simulation definitions). Note that while one step protocols cannot be zero-knowledge even with respect to the prespecified verifier V, that is not the case when considering two-step protocols. There exist two known such protocols which *are* zero-knowledge with respect to V (under all definitions) for languages believed not to be in BPP (Quadratic Non-Residuosity [GMR1], 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 exist a simulator M_{V^*}.

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 *iff* $x \in L$. When V follows the protocol, it "knows" the answer to its questions, but this is no longer guaranteed for arbitrary V^*'s. In fact, 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 which are zero-knowledge (with respect to any verifier) [GMR1, GMW1].

Theorem 6: Let L be a language for which there exists a two-step auxiliary input zero-knowledge proof system. Then $L \in BPP$.

4.3 Known 2-Step Interactive Proofs

In the previous subsection we proved a general result concerning 2-step protocols under the auxiliary-input definition of zero-knowledge. We believe that the above triviality result holds for the original [GMR1] definition too. In this subsection we only prove that two *specific* 2-step interactive proof are not zero-knowledge even under the [GMR1] definition, unless the corresponding languages are in BPP.

Let $QNR = \{(y,N) \mid y \in Z_N^*$ is a quadratic nonresidue with Jacobi symbol $1\}$.

The following is an interactive proof system for QNR, originating from [GMR1]:

V: If the Jacobi symbol of $y \bmod N$ is -1 then reject. For each i, $1 \le i \le n = |N|$, choose $r_i \in_R Z_N^*$ and $b_i \in_R \{0,1\}$ and set $t_i = r_i^2 \cdot y^{b_i}$.

$V \rightarrow P$: $t_1, t_2, ..., t_n$ (the notation $V \rightarrow P: m$ means V sends message m to P.)

P: If not all t_i's have Jacobi symbol 1 then notify "error" and stop.
For each i, set $\beta_i = 1$ if t_i is a quadratic residue and 0 otherwise.

$P \rightarrow V$: $\beta_1, \beta_2, ..., \beta_n$

V: If for all i $b_i = \beta_i$ then accept. Otherwise reject.

Theorem 7: If the above 2-step protocol for QNR is zero- knowledge, then $QNR \in BPP$.

Using a similar technique, we can show:

Theorem 8: If the [GMW1] 2-step protocol for Graph Non-Isomorphism (GNI) is zero-knowledge, then $GNI \in BPP$.

4.4 Zero Knowledge Proofs Which Never Err

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 $x \notin L$ (see [GMS] for formal definition). In this section we show that no protocol of this type can be zero-knowledge, even with respect to the [GMR1] definition.

Theorem 9: Let L be a language for which there exists a zero-knowledge Las Vegas interactive proof system. Then $L \in R$.

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

We show here that under the auxiliary-input definition of zero-knowledge, randomness of the prover is *essential* to the non-triviality of zero-knowledge proof systems. In other words, any language which has an auxiliary-input zero-knowledge proof system in which the prover is deterministic belongs to BPP.

Theorem 10: 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 sketch: Let $(\alpha_0, \beta_1, \alpha_1, ..., \beta_k, \alpha_k)$ be a text of the given protocol $<P,V>$, where the α_i's are the prover's messages and the β_i's are the verifier's messages (i.e. $\beta_i = V(x, r, \alpha_0, \ldots, \alpha_{i-1})$, where x is the input and r is V's coin tosses). The key to this proof is the fact that if P is deterministic, then every message it sends is a function of x and of all previous verifier messages. We first show that also in the simulation text the message sent on behalf of the prover is determined (with high probability) by the previous messages of the verifier. We consider a verifier V^* which on auxiliary input $\beta_1, \beta_2, ..., \beta_i$ uses β_1 through β_i as its first i messages to P. The BPP machine M will use M_{V^*} (V^*'s simulator) as follows: first choose a random string r, and run M_{V^*} with empty auxiliary input to obtain α_0 (the rest of the text is discarded). It will compute β_1 as V would, using r and α_0 and run M_{V^*} with the computed β_1 as auxiliary input, to obtain α_1 (again, the rest of the text is discarded). This process is repeated until all $k+1$ prover messages are obtained, and then M decides whether to accept or reject in the same way V would. It can be shown that if $x \notin L$ then M will accept with negligible probability (this follows from the fact that the original protocol is an interactive proof). It is left to show that if $x \in L$ then M will accept with very high probability. Let $\alpha_0', \beta_1, \cdots, \alpha_{i-1}', \beta_i$ be a prefix of the text produced by M_{V^*} on auxiliary input $\beta_1, \cdots, \beta_i$. Then, for every $0 \le j \le i-1$, with very high probability $\alpha_j' = \alpha_j$ and consequently $\beta_j = V(x, r, \alpha_0' \cdots \alpha_{j-1}')$ follows. Thus the conversation produced by M when choosing r equals (with high probability) the conversation between P and V when V uses r. Since r was chosen at random it follows from the completeness property of interactive proofs that M must accept with very high probability. □

ACKNOWLEDGEMENTS

I would like to thank Oded Goldreich for his extensive collaboration on this work. I would also like to thank my advisor Prof. Shimon Even for supervising this research and making useful comments on the paper. The concept of Las Vegas interactive proofs was raised by Manuel Blum and communicated through Silvio Micali and Oded Goldreich. 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.

[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.

[GMS] Goldreich, O., Y. Mansour, and M. Sipser, "Interactive Proof Systems: Provers that Never Fail and Random Selection", these proceedings.

[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, "Knowledge Complexity of Interactive Proofs", to appear in *Siam J. on Comp.*.

[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.

[O] Oren, Y., M.Sc. Thesis, in preparation.

[TW] Tompa, M., and H. Woll, "Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information", these proceedings.

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