A Knowledge-Based Analysis of Zero Knowledge

(Preliminary Report)

Joseph Y. Halpern
IBM Almaden Research Center
San Jose, CA 95120

Yoram Moses
Weizmann Institute
Rehovot, 76100
Israel

Mark R. Tuttle
MIT Laboratory for Computer Science
Cambridge, MA 02139

Abstract: While the intuition underlying a zero knowledge proof system [GMR85] is that no "knowledge" is leaked by the prover to the verifier, researchers are just beginning to analyze such proof systems in terms of formal notions of knowledge. In this paper, we show how interactive proof systems motivate a new notion of *practical knowledge*, and we capture the definition of an interactive proof system in terms of practical knowledge. Using this notion of knowledge, we formally capture and prove the intuition that the prover does not leak any knowledge of any fact (other than the fact being proven) during a zero knowledge proof. We extend this result to show that the prover does not leak any knowledge of how to compute any information (such as the factorization of a number) during a zero knowledge proof. Finally, we define the notion of a *weak interactive proof* in which the prover is limited to probabilistic, polynomial-time computations, and we prove analogous security results for such proof systems. We show that, in a precise sense, any nontrivial weak interactive proof must be a proof about the prover's knowledge, and show that, under natural conditions, the notions of interactive proofs of knowledge defined in [TW87] and [FFS87] are instances of weak interactive proofs.

The work of the second author was supported in part by a Sir Charles Chlore fellowship. The work of the third author was supported in part by the Office of Naval Research under Contract N00014-85-K-0168, by the National Science Foundation under Grants DCR-83-02391 and CCR-8611442, and by the Defense Advanced Research Projects Agency (DARPA) under Contract N00014-83-K-0125. This paper appears in *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988, pp. 53–65.

1 Introduction

The notions of *interactive proof* and *zero knowledge*, introduced by Goldwasser, Micali, and Rackoff in [GMR85], have been the subject of extensive research (see, for example, [BC86, FFS87, For87, GHY85, GMW86, GS87, Ore87, TW87]). Informally, an interactive proof is a two-party conversation in which an infinitely powerful "prover" tries to convince a polynomial-time "verifier" of the truth of some fact φ (typically of the form $x \in L$) through a sequence of interactions. Roughly speaking, such an interactive proof is said to be zero knowledge if, whenever φ holds, the verifier is able to generate on its own the conversations it could have had with the prover during an interactive proof of φ. Intuitively, the verifier does not learn anything from such conversations with the prover (other than the fact φ) that it could not have learned on its own by generating these conversations itself. Consequently, the only knowledge gained by the verifier during an interactive proof is that which the prover initially set out to prove.

This informal discussion makes it quite clear that our intuition concerning interactive proofs and zero knowledge is intimately related to a notion of knowledge. While this intuition is quite compelling, it is based on an operational notion of *being able to generate* whose formal relationship to knowledge is not immediately obvious. It is this relationship that is the focus of our paper.

The formal notions of knowledge needed to capture our understanding of interactive proofs and zero knowledge are far more subtle than the standard information-theoretic notion of knowledge that has been used successfully in the analysis of distributed systems (see, for example, [CM86, DM86, FI86, Had87, HM84, HZ87, LR86, MT88, NT87, PR85]; see [Hal87] for an overview). Since both probability and the computational power of the prover and verifier play crucial roles in the definition of zero knowledge, the notions of knowledge used to reason about zero knowledge must take these issues into account. It is relatively straightforward to extend the standard notion of knowledge to include probability. Our definition of probabilistic knowledge is based on definitions given by Fagin and Halpern in [FH88]. Dealing with complexity, however, is more difficult. The approach we use is based on the notion of resource-bounded knowledge introduced by Moses in [Mos88]. While a number of extensions of resource-bounded knowledge are possible, we believe ours are well-suited to the context of cryptographic protocols, as well as other contexts.

An issue related to the notion of knowing a fact is that of knowing how to perform various operations. For example, there is a difference between knowing *the fact* that a certain number is a product of two primes and knowing *how to generate* the two prime factors. Zero knowledge proofs are not intended to leak any knowledge of this kind as well as any knowledge of facts. While this notion of "knowing how" has also been of great interest in Philosophy and AI (see [Moo85]), standard notions of knowledge do not capture this aspect of knowledge. To do so, we define a notion of knowing how to generate a y satisfying a relation $R(x, y)$, within given resource bounds.

The main contributions of this paper are:

- We capture the definition of an interactive proof using knowledge and probability. We then show how interactive proofs motivate the definition of *practical knowledge*, and capture the definition of an interactive proof in terms of practical knowledge.
- Using practical knowledge, we prove that with high probability the verifier in a zero knowledge proof of $x \in L$ knows a fact φ at the end of the proof iff it knows $x \in L \supset \varphi$ at the beginning of the proof. Intuitively, this captures the idea that zero knowledge proofs do not "leak" knowledge of facts other than those that follow from $x \in L$.
- We define a notion of knowing how to generate a y satisfying a relation $R(x, y)$, and prove that with high probability if the verifier in a zero knowledge proof of $x \in L$ knows how to generate a y satisfying $R(x, y)$ at the end of the proof, then it knows how to do so at the beginning as well. This captures our intuition that at the end of a zero knowledge proof the verifier can not do anything that it could not do at the beginning.
- We consider *weak interactive proofs*, in which the prover is restricted to probabilistic, polynomial-time computations (and hence is no longer infinitely powerful). This is the most relevant context in practice. While interesting interactive proofs for membership in a language do exist (see [GMR85, GMW86]), we prove that any language L having a *weak* interactive proof is contained in BPP, and hence the verifier can determine whether $x \in L$ on its own without consulting the prover. Consequently, we are led to consider weak interactive proofs about the prover's initial state (that is, the prover's knowledge) rather than weak interactive proofs of language membership. We then proceed to show that, under natural conditions, the notions of interactive proofs of knowledge defined in [FFS87] and [TW87] are instances of weak interactive proofs. Finally, we show that zero knowledge weak interactive proofs guarantee the same type of security with respect to the facts they prove as zero knowledge interactive proofs guarantee with respect to language membership.

We believe that this analysis provides a great deal of insight into (and support for) the definitions in [GMR85] and their extensions to the case of proofs about knowledge in [FFS87, TW87]. None of the technical results is very deep; the difficulty was in coming up with the right notions. We believe that our viewpoint provides a good framework in which to think about these definitions and their appropriateness.

Fischer and Zuck [FZ87] also consider notions of knowledge appropriate for cryptographic protocols. They give definitions that are related to the ones given here (we discuss the relationship in the full paper) and use their definitions to analyze an interactive proof of quadratic residuousity.

The rest of the paper is organized as follows. In the next section, we briefly review how to ascribe knowledge to processors in a distributed system, how to extend the standard definition of knowledge to include probability, and how to account for bounds on the processors' computational resources. In Section 3 we review the definition of an interactive proof, and capture this definition in terms of probabilistic knowledge. In Section 4 we show how interactive proofs motivate the definition of *practical knowledge*, and capture the definition of an interactive proof system in terms of practical knowledge. In Section 5 we define zero knowledge. This definition of zero knowledge is a weaker definition than that given in [GMR85], and hence our results will hold for [GMR85] as well. In Section 5.1 we make precise (using knowledge) the intuition that in a zero knowledge proof the verifier does not know any more at the end than it did at the beginning. In Section 5.2 we define the notion of "knowing how," and show that, in a precise sense, the verifier cannot *do* any more at the end of a zero knowledge proof than it could at the beginning. Finally, Section 6 introduces weak interactive proofs, relates them to the proofs of knowledge of [FFS87, TW87], and proves that zero knowledge weak interactive proofs are secure in the senses defined above.

2 Knowledge

Our analysis of interactive and zero knowledge proof systems depends heavily on the definitions of knowledge, probabilistic knowledge, and resource-bounded knowledge. We now briefly review these definitions. Since we will be considering only probabilistic protocols running in synchronous systems, we will restrict the generality of these definitions to this context.

We begin with a sketch of our formal model of computation (cf. [DM86, MT88]). The distributed systems we consider consist of a finite collection of processors, each pair of which is connected by a two-way communication link. Processors share a global clock that starts at time 0 and proceeds in discrete increments of one. Computation in the system proceeds in rounds, round m lasting from time $m-1$ to time m. During a round, every processor first performs some (possibly probabilistic) local computation, then sends messages to other processors, and then receives all messages sent to it during the round. Each processor begins with some initial local state at time 0. At any given time, a processor's local state consists of the current time on the global clock, its initial state, the history of messages it has received from other processors, and the history of coin flips it has used. A global state is a tuple of local states, one for each processor. We think of each processor as following a *protocol* which specifies in any given state what messages each processor is required to send as well as what actions it is required to perform as a (possibly probabilistic) function of the processor's local state. An infinite execution of such a protocol (an infinite sequence of global states) is called a *run*. We define a *system* to be a set of such runs, often the set of all possible runs of a particular protocol. Given a run r and a time m, we refer to (r, m) as a *point*, and we say that (r, m) is a point of the system $\mathcal{R}$ if $r \in \mathcal{R}$. We denote the global state at the point (r, m) (that is, the global state at time m in r) by $r(m)$, and we denote the local state of processor q in $r(m)$ by $r_q(m)$.

We now turn to the standard definition of knowledge in distributed systems (cf. [CM86, FI86, HF85, HM84, PR85]). The intuition behind this definition is that a processor can be said to *know* that a fact φ is true if, based on its information about the current state of affairs, φ must be true. Roughly speaking, therefore, the processor q is said to *know* φ at a point (r, m) if φ is true of all points q considers possible at (r, m). Since a processor's knowledge depends on the set from which the points it considers possible may be chosen, a processor's knowledge is always defined with respect to a system $\mathcal{R}$. We assume that we have a collection of primitive facts about the system, facts such as "*the value of the variable x is a prime number*" that do not involve the processors' knowledge. Each fact φ is identified with a set $\pi(\varphi)$ of points interpreted as the set of points at which φ holds. We say that a point (r, m) in a system $\mathcal{R}$ *satisfies* φ, which we denote by $(\mathcal{R}, r, m) \models \varphi$, if $(r, m) \in \pi(\varphi)$. We often write $(r, m) \models \varphi$ rather than $(\mathcal{R}, r, m) \models \varphi$ when the system $\mathcal{R}$ is clear from context. We say that a fact φ is *valid in the system* $\mathcal{R}$, which we denote by $\mathcal{R} \models \varphi$, if $(r, m) \models \varphi$ for all points (r, m) of $\mathcal{R}$. We extend this collection of primitive facts to a logical language by closing under the usual boolean connectives and several modal operators. Two modal operators included are the linear temporal logic operators $\Box$ and $\Diamond$: the formula $\Box\varphi$ holds at a point (r, m) iff the formula φ holds at all points (r, m') with $m' \geq m$; and $\Diamond\varphi$ holds at (r, m) iff the formula φ holds at some point (r, m') with $m' \geq m$. Most interesting, however, are the modal operators K_q, one for every processor q, where the formula $K_q\varphi$ is read "*q knows φ*." As previously indicated, $K_q\varphi$ is defined as follows: $(r, m) \models K_q\varphi$ iff $(r', m) \models \varphi$ for all points (r', m) of $\mathcal{R}$ such that $r_q(m) = r'_q(m)$. A processor therefore knows precisely those facts that follow from the information contained in its local state.

We are often interested in facts that, although they do not hold in *all* the states that a processor considers possible, do hold at a certain fraction of those points. When reasoning about probabilistic systems, it is important to be able to make statements such as "according to q, the fact φ holds with probability α." Intuitively, such a statement might mean that α is the conditional probability of φ, given q's local state. Fagin and Halpern formalize and generalize this intuition in [FH88] as follows. Given a system $\mathcal{R}$, they associate with every processor q and every point $c = (r, m)$ a probability space $\mathcal{P}(q, c) = (S_{q,c}, X_{q,c}, \mu_{q,c})$, where $S_{q,c}$ is a set of points, $X_{q,c}$ is a set of measurable subsets of $S_{q,c}$, and $\mu_{q,c}$ is a probability measure. Intuitively, the set $S_{q,c}$ is a subset of the points q thinks possible at c, and $\mu_{q,c}$ determines the probability with which q considers a particular point in $S_{q,c}$ to be the actual point c. The set $S_{q,c}(\varphi)$ is then defined to consist of those points in $S_{q,c}$ at which φ holds. In order to reason about probability, we allow formulas of the form $Pr_q(\varphi) \geq \alpha$, with semantics defined by $c \models Pr_q\varphi \geq \alpha$ iff $\mu_{q,c}(S_{q,c}(\varphi)) \geq \alpha$.[1] We define $K_q^\alpha\varphi$ to be an abbreviation for $K_q(Pr_q(\varphi) \geq \alpha)$, which intuitively says that processor q knows that φ must hold with probability at least α. Finally, we define $E^\alpha\varphi$ to be an abbreviation for $\wedge_q K_q^\alpha\varphi$, where the conjunction is taken over all the processors in the system. Intuitively, $E^\alpha\varphi$ means that *every* processor knows that φ must hold with probability α.

This definition of probabilistic knowledge leaves open a great deal of flexibility in the choice of probability spaces $\mathcal{P}(q,c)$. While it may seem at first that the correct choice for $S_{q,c}$ is the set of all points q is unable to distinguish from c, the analysis in this paper has shown that, due to the subtle interaction between the nondeterministic and probabilistic choices made by processors, there are good reasons to choose $S_{q,c}$ to be a subset of these points. Consider, for example, the system determined by two processors q and q' running the following one-round protocol: processor q' starts with a one-bit initial state, flips a fair coin, and performs a particular action a iff the outcome of the coin toss is equal to the bit in its initial state. Clearly, this system consists of four runs of the form $r_{b,f}$, where b is the value of the bit in q''s initial state and f is the outcome of the coin flip. Now let us consider the probability with which q knows at time 0 that q' will perform a. Suppose that with every time 0 point we associate the probability space consisting of all time 0 points (notice that q considers all such points to be possible at time 0). Since the only probability distribution we have is on the coin flipped by q', the only nontrivial events to which we can assign a probability are $\{(r_{0,h},0),(r_{1,h},0)\}$ ("the coin lands heads") and $\{(r_{0,t},0),(r_{1,t},0)\}$ ("the coin lands tails"); we cannot assign a probability to the event "q' performs a"! Suppose, on the other hand, that with every time 0 point we associate the probability space of time 0 points having the same initial global state; that is, we can associate with a point of the form $(r_{b,f},0)$ the probability space $\{(r_{b,h},0),(r_{b,t},0)\}$. In this probability space, we assign each of $\{(r_{b,h},0)\}$ and $\{(r_{b,t},0)\}$ probability $1/2$. Consequently, at every point q considers possible at time 0, the probability according to q that q' performs a is $1/2$, and it follows that q knows with probability $1/2$ that q' performs a, as we would expect. (This example is discussed in greater detail in [FH88].)

This observation leads us to choose the probability space associated with the processor q and point $c = (r,m)$ as follows. We take $S_{q,c}$ to be the set of points (r',m) such that $r'(m) = r(m)$ (that is, the set of points (r',m) having the same global state as (r,m)). Notice that in our model, given a global state at time m, each run having that global state at time m is determined by the sequence of coins flipped after time m in the run. Consequently, each set $S_{q,c}(\varphi)$ can be identified with a set of coin flips, which in this paper will always be measurable. The probability measure $\mu_{q,c}$ therefore assigns to the event $S_{q,c}(\varphi)$ the probability of the set of coin flips identified with $S_{q,c}(\varphi)$. Notice that having made this choice of probability spaces, the operators Pr_q are identical for all q, and hence we will omit subscripts in the remainder of this work.[2]

Returning to the standard definition of knowledge, notice that a processor is said to know all facts that follow from its local state, regardless of the computational complexity of determining that these facts hold. When analyzing cryptographic protocols, where the computational intractability of a problem is used to keep secret certain pieces information, such a notion of knowledge is clearly inappropriate. In [Mos88], Moses introduces a notion of *resource-bounded knowledge* that takes into account bounds on a processor's computational resources. The intuition behind this notion is that the only way a resource-bounded processor can know a fact is if it can compute that it knows this fact. In this work we are concerned with the facts a processor can compute using a probabilistic test running in time polynomial in some parameter depending on its current state (usually that parameter will be $|x|$, where x is the common input). Thus, we consider only *BPP knowledge*, which seems most appropriate in the context of interactive proofs. Given a system $\mathcal{R}$, a probabilistic algorithm M is said to be a *BPP test for* $K_q\varphi$ in $\mathcal{R}$ if, for all points (r,m) of $\mathcal{R}$, M's computation starting from $r_q(m)$ runs in time polynomial in $|x|$, accepts with probability at least $2/3$ if $(r,m) \models K_q\varphi$, and rejects with probability at least $2/3$ if $(r,m) \not\models K_q\varphi$. We say that q *BPP-knows* φ at a point (r,m) of $\mathcal{R}$, denoted by $(r,m) \models K_q^{\mathrm{BPP}}\varphi$, iff

[1] Fagin and Halpern actually write $m_q(\varphi) \geq \alpha$ instead of $Pr_q(\varphi) \geq \alpha$, and in fact define a much richer language than we do here. They also show how to deal with the possibility of nonmeasurable sets. As our sets $S_{q,c}(\varphi)$ will always be measurable, we omit these details here.

[2] We note that there are at least two notions of probabilistic knowledge that seem relevant in the context of cryptographic protocols, the one given here and another outlined in [FZ88] (the spirit of which can be captured in the framework of [FH88]). Each has its own philosophical advantages, and we refer the reader to [FH88, FZ88] for extended discussions. However, since the definitions of interactive proofs and zero knowledge state conditions on the objective probability of events at time 0, we will be concerned only with a processor's probabilistic knowledge at time 0, and at time 0 the two definitions of probabilistic knowledge coincide. We have chosen the definition outlined here since the fact that the probability spaces are independent of the agent simplifies our analysis slightly.

$(r,m) \models K_q\varphi$ and there is a BPP test for $K_q\varphi$ in $\mathcal{R}$. Thus, a processor BPP-knows φ if it knows φ and there is a BPP algorithm with which it can compute that it knows φ. (Of course, there is nothing special about the values 2/3 used in the definition of BPP tests. We could have used any value bounded away from and above 1/2.)

Similar notions of knowledge can be defined with respect to other complexity classes. We refer the reader to [Mos88] for a detailed discussion of a number of interesting properties of these notions of knowledge.

3 Interactive Proof Systems

In this section, we first review the notion of an interactive proof system (our definitions are essentially those of [GMR85]), and then show that the definition of an interactive proof system can be captured in terms of probabilistic knowledge.

An *interactive protocol* is an ordered pair (P,V) of probabilistic Turing machines. P and V share a read-only *input tape*; each has a private one-way, read-only *random tape*; each has a private *work tape*; and P and V share a pair of one-way *communication tapes*, one from P to V being write-only for P and read-only for V, and the other from V to P being write-only for V and read-only for P. A *run* of the protocol (P,V) is defined as follows. To begin with, the input tape is initialized with some common input, say x; each random tape is initialized with an infinite sequence of random bits; each work tape may or may not be initialized with an initial string;[3] and the communication tapes are initially blank. The run then proceeds in a sequence of rounds. During any given round, V first performs some internal computation making use of its work tape and other readable tapes, and then sends a message to P by writing on its write-only communication tape; P then performs a similar computation. Either P or V may halt the interaction at any time by entering a halt state. V *accepts* or *rejects* the interaction by entering an accepting or rejecting halt state, respectively, in which case we refer to the resulting run as either an accepting or rejecting run. The running time of P and V during a run of (P,V) is the number of steps taken by P and V, respectively, during the run. We assume that V is a probabilistic Turing machine running in time polynomial in $|x|$, and hence that it can perform only probabilistic, polynomial-time computations during each round. For now we make no assumptions about the running time of P, although in Section 6 we shall restrict attention to probabilistic, polynomial-time provers.

The system corresponding to runs of the interactive protocol (P,V) can be described in terms of the computational model defined in Section 2 as follows. The system consists of two processors, p and v, running the protocols P and V, respectively. A run is an infinite sequence of global states, where each global state consists of a local state for each of p and v. Processor p's local state is a tuple consisting of a description of the Turing machine P, the current round number, the contents of the input tape, the finite prefix of its random tape read up to this point, the contents of its work tape, the contents of the two communication tapes, and the position of the tape heads on each of these tapes; processor v's local state is defined in a similar fashion. We denote by $P \times V$ the system consisting of all possible runs of (P,V), by $P \times \mathcal{V}^{pp}$ the system consisting of the union of the systems $P \times V^*$ for all probabilistic, polynomial-time V^*, by $\mathcal{P} \times V$ the system consisting of the union of the systems $P^* \times V$ for all Turing machines P^*, and by $\mathcal{P}^{pp} \times V$ the system consisting of the union of the systems $P^* \times V$ for all probabilistic, polynomial-time P^*. Note that we distinguish p and v, the "prover" and the "verifier" respectively, from the protocols that they are running. In the system $P \times V$, the verifier is always running the same protocol (namely V) in all runs. In the system $P \times \mathcal{V}^{pp}$, the verifier may be running different protocols in different runs.

Let us denote by $(P(s),V(t))(x)$ the random variable assuming as values the runs of (P,V) (according to the probability distribution generated by the protocol (P,V)) in which the prover's work tape is initialized with s, the verifier's work tape is initialized with t, and the input tape is initialized with x. An interactive protocol (P,V) is said to be an *interactive proof system for a language* L if the following conditions are satisfied:

- *Completeness*: For every k and sufficiently large x, and for every s and t, if $x \in L$ then

$$Pr\left[(P(s),V(t))(x) \text{ accepts}\right] \geq 1 - |x|^{-k}.$$

- *Soundness*: For every k and sufficiently large x, for every P^*, and for every s and t, if $x \notin L$ then

$$Pr\left[(P^*(s),V(t))(x) \text{ accepts}\right] \leq |x|^{-k}.$$

We refer to p as the "good prover" when it is running P, and to v as the "good verifier" when it is

[3] The motivation for allowing initial values on the verifier's and prover's work tapes can be found in [Ore87, TW87]. Allowing initial information on the prover's worktape is particularly important in the case of resource-bounded provers considered in Section 6.

running V. The completeness condition is a guarantee to both the good prover and the good verifier that if $x \in L$, then with overwhelming probability the good prover will be able to convince the good verifier that $x \in L$. The soundness condition is a guarantee to the good verifier that if $x \notin L$, then the probability that an arbitrary (possibly malicious) prover is able to convince the good verifier that $x \in L$ is very low.

Notice that in our soundness condition, the "sufficiently large x" depends only on the value of k, and not on the choice of P^*. In the original definition of interactive proof given in [GMR85], it is not clear whether the dependence is on k only or on both k and P^*. As Shafi Goldwasser pointed out to us, in the case of infinitely powerful provers, it doesn't matter what choice we make. (More formally, an interactive proof system (P, V) is sound with respect to one choice iff it is sound with respect to the other; we prove this in the full paper.) However, the choice does make a difference in the case of resource-bounded provers, as we shall see in Section 6.

We can translate these completeness and soundness conditions immediately into statements about probability in our language as follows. Let *init* be the fact holding only at points at the beginning of a run, and let *accept* be the fact holding only at points at which the verifier has accepted.

Proposition 1: An interactive protocol (P, V) is an interactive proof system for a language L iff the following conditions are satisfied:

- *Completeness*: For every k there exists c such that
$$P \times V \models \mathit{init} \supset Pr[x \in L \supset \Diamond \mathit{accept}] \geq 1 - c\,|x|^{-k}.$$
- *Soundness*: For every k there exists c such that
$$\mathcal{P} \times V \models \mathit{init} \supset Pr[\Diamond \mathit{accept} \supset x \in L] \geq 1 - c\,|x|^{-k}.$$

The constant c above is necessary due to the fact that the probabilistic guarantees made by the definition of an interactive proof system hold only for "sufficiently large x." Notice that if $1 - c\,|x|^{-k}$ is negative, then $Pr(\varphi) \geq 1 - c\,|x|^{-k}$ is equivalent to $Pr(\varphi) \geq 0$, which is valid for every fact φ. Consequently, by choosing c so that $1 - c\,|x|^{-k} < 0$ for insufficiently large x we obtain a formula holding for all x, and hence valid at all points of the system. While this constant c does not appear in the formal definition of an interactive proof system, an equivalent definition of interactive proof systems can be formulated making use of such constants just as we do in Proposition 1.

Notice that, according to Proposition 1, a formula such as $Pr(x \in L \supset \Diamond \mathit{accept}) \geq 1 - c\,|x|^{-k}$ holds at time 0 but not necessarily at later points. After the verifier has rejected, for example, it is clearly not the case that with high probability the verifier will eventually accept. In general, even before the verifier has actually decided to accept or reject, a particularly bad sequence of coin flips can significantly lower the verifier's chances of eventually accepting. Intuitively, this is due to the fact that the verifier's probability space is changing with every step. (In our case, the probability space we associate with a point is the set of points having the same global state, a set that decreases in size with every step.) Consequently, the antecedent *init* is crucial in the formulas above.

Since the facts appearing in Proposition 1 are valid, all processors know these facts at all points. Furthermore, all processors know the fact *init* whenever it holds. Since from $K_q \mathit{init}$ and $K_q(\mathit{init} \supset \psi)$ we can deduce $K_q \psi$, we can deduce the following corollary to Proposition 1.

Corollary 2: An interactive protocol (P, V) is an interactive proof system for a language L iff the following conditions are satisfied:

- *Completeness*: For every k there exists c such that
$$P \times V \models \mathit{init} \supset E^{1-c|x|^{-k}}(x \in L \supset \Diamond \mathit{accept}).$$
- *Soundness*: For every k there exists c such that
$$\mathcal{P} \times V \models \mathit{init} \supset K_v^{1-c|x|^{-k}}(\Diamond \mathit{accept} \supset x \in L).$$

In other words, (P, V) is complete if both the good prover and the good verifier know with high probability that if $x \in L$, then the good prover will convince the good verifier to accept; and (P, V) is sound if the good verifier knows with high probability that, no matter what protocol the prover is running, if the verifier accepts x then $x \in L$. It is actually the case that if (P, V) is sound then every prover also knows with high probability that if the verifier accepts x then $x \in L$ (that is, we could have replaced $K_v^{1-c|x|^{-k}}$ by $E^{1-c|x|^{-k}}$ in the case of soundness above); we have chosen this formulation since it is the good verifier's knowledge that is essential in the context of soundness.

4 Practical Knowledge

As we have just seen, the definition of an interactive proof system can be characterized in terms of probabilistic knowledge. This characterization, however, is a slight reformulation of the original definition in terms of very similar concepts. It does not capture, for example, the intuition that at the end of an interactive proof of $x \in L$ with the good prover, the good verifier knows that $x \in L$ despite its limited computational power. In this section we show how the definition of an interactive proof system motivates a new notion of *practical knowledge* which will enable us to formalize this intuition. In later sections, practical knowledge will play a crucial role in capturing the security provided by zero knowledge proof systems.

We have already argued that the notion of resource-bounded knowledge introduced in [Mos88] seems to be a natural way of capturing the knowledge of a resource-bounded processor, with BPP knowledge being most relevant to the cryptographic setting. Unfortunately, there is a notion of "learning" of great importance to cryptographic protocols (and, in particular, to interactive proof systems) that can not be captured directly in terms of BPP knowledge. Consider, for example, a fact φ such as "*the input x has a factor smaller than $\sqrt[5]{x}$*". At a point in which the processor has the factorization of x available to it, say the factorization happens to be written on its work tape, and x has a small factor, we might like to say that the processor knows φ despite its limited resources. Recall that a processor BPP-knows φ iff it knows φ and it has a BPP test for φ. Assuming that there is no BPP test for φ, our processor will *never* BPP-know φ. This is true even when it has the factorization written on its work tape. Thus, a naive use of the notion of BPP knowledge does not seem to allow us to capture the idea of learning.

Notice, however, that when a processor finds the factorization of x on its work tape and hence is able to determine that φ holds, the processor learns a great deal more than just the fact φ. It actually learns a fact ψ that implies φ, where ψ is the fact "*the factorization of x is on the work tape, and it contains a factor smaller than $\sqrt[5]{x}$*". Since this fact ψ is clearly BPP testable, the processor actually BPP-knows ψ. In other words, the processor learns φ as a result of coming to BPP-know a stronger fact ψ that implies φ. In this sense, the notion of learning φ *can* be captured in terms of BPP knowledge.

At this point, one might be tempted to define a notion of learning in which a processor learns φ at a point if at this point it BPP-knows a fact ψ that implies φ. Unfortunately, this notion of learning is not very useful to a resource-bounded processor. It could be, for example, that at every point the processor BPP-knows a different fact ψ implying φ (and hence has "learned" φ everywhere) and yet is unable to determine at a particular point which fact ψ it should test for in order to determine that it knows φ. Alternatively, one might be tempted to define a notion of knowing φ with respect to a particular test M where, informally, a processor knows φ with respect to M if with the test M the processor can determine that it knows φ. We consider such notions to be unsatisfactory, however, since if knowledge is to be used for protocol specification it must be possible to abstract the particular tests being used. We note that the definition of resource-bounded knowledge already existentially quantifies over such tests (so these tests do not appear in the notation used), and we do not want to reintroduce them here.

We take an alternative (and more direct) approach to capturing learning. The idea is to define a notion of BPP knowledge of φ relative to a set A of points. We can think of the points of A as the points at which a processor learns φ. Roughly speaking, this means that we have a BPP test for $K_q\varphi$ that correctly determines whether q knows φ at all points of A, but may only satisfy weaker requirements off A (in particular, the test may not be required to accept with high probability at points off A at which $K_q\varphi$ holds).[4]

More formally, we proceed as follows. We say that a test M is *sound* for φ at a point (r,m), denoted by $(r,m) \models \mathit{sound}(M,\varphi)$, if $(r,m) \not\models \varphi$ implies that M rejects at (r,m) with probability at least 2/3. Similarly, we say that M is *complete* for φ at (r,m), denoted by $(r,m) \models \mathit{complete}(M,\varphi)$, if $(r,m) \models \varphi$ implies that M accepts at (r,m) with probability at least 2/3. Given a system $\mathcal{R}$, a set A of points in $\mathcal{R}$, and a fact φ, we say that "*q BPP-knows φ with respect to A*" at a point (r,m), denoted by $(r,m) \models K_q^{\mathrm{BPP},A}\varphi$, iff

1. $(r,m) \in A$,
2. $(r,m) \models K_q\varphi$, and
3. there is a probabilistic Turing machine M taking as input the local state of q and running in time polynomial in $|x|$ (where x is the common input in the global state $r(m)$) such that
   (a) M is a *sound* test for $K_q\varphi$ on $\mathcal{R}$; that is, $\mathcal{R} \models \mathit{sound}(M, K_q\varphi)$.
   (b) M is a *complete* test for $K_q\varphi$ on A; that is, $\mathcal{R} \models$ 'in A' $\supset \mathit{complete}(M, K_q\varphi)$ where 'in A' is the fact holding at precisely those points in A.

[4] We remark that Fischer and Zuck define in [FZ87] a notion of knowing with respect to a Turing machine M which is intended to deal with some of the same problems as our notion of knowing with respect to a set A. We compare our definitions with those of [FZ87] in detail in the full paper.

The first condition is a technical one, which ensures that knowledge with respect to A can hold only at points of A. We restrict attention to points in A since these are the only points of interest (and the only ones where our test is guaranteed to be correct). The second condition requires that φ actually be known, as is required by BPP knowledge. The third condition requires the existence of a test for φ that is sound on $\mathcal{R}$ and complete on A.

It is clear that a processor BPP-knows φ iff it BPP-knows φ with respect to the set of all points in $\mathcal{R}$, and hence that the definition of BPP knowledge with respect to a set of points is a direct generalization of BPP knowledge. In fact, it is easy to see that if the fact 'in A' is testable in BPP, then $K_q^{\mathrm{BPP},A}\varphi$ is equivalent to $K_q^{\mathrm{BPP}}(\text{'in } A\text{'} \wedge \varphi)$. Furthermore, notice that we are now able to capture the notion of a processor learning a fact φ as the result of its unexpected acquisition of information. Returning to our example, let A be the set of points where q has the factorization of x on its work tape. Let M be the test that rejects if the factorization of x is not on the work tape or if the factorization is on the worktape and there are no small factors. This test M for φ is clearly sound everywhere and complete on A. Thus, when q learns from the factorization of x on its work tape that φ must be true, q knows φ with respect to the set A.

The reader may wonder at the asymmetry of our definition. Why do we require soundness on all of $\mathcal{R}$, but completeness only on A? Notice that if we strengthen the definition to require soundness and completeness on all of $\mathcal{R}$, then we have essentially returned to the definition of BPP knowledge. On the other hand, suppose we weaken the definition to require soundness only on A. If membership in the set A is easily testable, then such a notion of knowledge may be of interest. We will, however, be forced to consider arbitrary sets A in this work, and in this context it becomes rather uninteresting. For suppose that processor q has access to an algorithm M that is guaranteed to be sound and complete only on A. Moreover, suppose that when q runs M repeatedly on its state $r_q(m)$ at some point (r, m), it finds that M almost always accepts. In this case, q knows that *if* it is at a point in A, then φ holds. But since it may be quite difficult for q to determine whether it is at a point in A, this may not be very useful information. With our definition, q would know that $K_q\varphi$ (and hence φ) holds at this point, regardless of whether the point is in A (since, by our definition, if $\neg K_q\varphi$ holds, then M rejects with high probability). Of course, if M almost always rejects on input $r_q(m)$, then q can say nothing without knowing whether (r, m) is in A. We could instead have required completeness on all of $\mathcal{R}$ and soundness on A. In cryptographic applications, however, it tends to be more important to be able to learn that φ is true than to learn that it is false. This choice is a matter of taste.

Let us return, now, to the context of interactive proof systems (P, V) for L. Let us say that a point of the system $P \times V$ is a *final point* if at that point the verifier has either accepted, rejected, or otherwise halted. Consider the set A of final points of $P \times V$, and consider the test M that accepts at a point if the verifier has accepted at that point and rejects otherwise. Intuitively, we would like to say that if $x \in L$, then the good verifier BPP-knows $x \in L$ with respect to A at the end of a proof of $x \in L$ with the good prover. Unfortunately, the test M is not a sound test for $x \in L$ since on rare occasions the verifier may incorrectly accept when $x \notin L$. In the context of probabilistic computations, however, a test that fails on a negligible portion of the cases is practically as good as one that never fails. Since the soundness and completeness conditions required by the definition of knowledge with respect to a set A do not allow for such freedom, we are led to the following notion of practical knowledge where these conditions are somewhat relaxed. Practical knowledge plays an important role in our analysis of interactive proofs and zero knowledge.

Recall that M is a sound test for φ in $\mathcal{R}$ if it is a sound test for φ at all points in the system $\mathcal{R}$. We would now like to consider such tests M that are sound tests for φ at all points of most runs of $\mathcal{R}$. Formally, we say that M is a *practically sound* test for φ if for all k there exists c such that

$$\mathcal{R} \models \mathit{init} \supset \\ Pr(\Box \mathit{sound}(M, K_q\varphi)) \geq 1 - c\,|x|^{-k}\,.$$

Similarly, if A is a set of points in $\mathcal{R}$, we say that M is a *practically complete* test for $K_q\varphi$ on A if for all k there exists c such that

$$\mathcal{R} \models \mathit{init} \supset \\ Pr(\Box[\text{'in } A\text{'} \supset \mathit{complete}(M, K_q\varphi)]) \geq 1 - c\,|x|^{-k}\,.$$

Notice that we have ensured that the probabilities used in defining practical soundness and practical completeness are taken at the beginning of the run through the use of the antecedent *init*. This means that we are effectively considering tests that behave correctly on all but a small fraction of the runs. We could have instead considered tests with the stronger property that they behave correctly at all but a small fraction of the points considered possible at any point (by deleting the antecedent *init*). This latter notion can lead to dramatically different results, but does not seem appropriate for most computer science applications (in particular, it is not appropriate for capturing interactive proofs).

We now define "*q practically BPP-knows φ* with respect to A" at a point (r,m), which we denote by $(r,m) \models \tilde{K}_v^{\mathrm{BPP},A}\varphi$, in precisely the same way as we defined "q BPP-knows φ with respect to A," except that the soundness and completeness conditions are replaced by practical soundness and practical completeness. This notion of knowledge may at first seem rather strange. Most previously defined notions of knowledge based on, say, polynomial-time tests have said that a processor knows φ at a point if its test for φ says that it knows φ. Here, however, since the tests allowed by the definition of practical knowledge may be in error on a small fraction of the runs, it is possible for a processor to have practical knowledge of φ with respect to A at a point in A even though its test for knowledge of φ may not indicate that it knows φ. When a processor practically knows φ with respect to A, it knows φ and has a test that quite accurately approximates this knowledge on the set A.

Now, returning to the problem of capturing the intuition that at the end of a proof of $x \in L$ with the good prover the good verifier learns $x \in L$, let us reconsider the set A and test M defined above: let A be the set of final points of $P \times V$, and let M be the test that accepts at a point if the verifier has accepted at that point and rejects otherwise. Notice that while the test M for φ is not sound everywhere and not complete on A, it is *practically sound* everywhere and *practically complete* on A. As a consequence, we have the following. We denote by 'p running P' the fact holding at a point iff at that point the prover is running the protocol P.

Proposition 3: If (P,V) is an interactive proof system for L, then

$$\mathcal{P} \times V \models (x \in L \wedge \text{`}p \text{ running } P\text{'}) \supset \Diamond \tilde{K}_v^{\mathrm{BPP},A}(x \in L),$$

where A is the set of final points of $\mathcal{P} \times V$ satisfying 'p running P'.

In fact, we can essentially prove the converse of this proposition as well, which shows that we can characterize the notion of an interactive proof system using practical knowledge.

Proposition 4: If

$$\mathcal{P} \times V^* \models (x \in L \wedge \text{`}p \text{ running } P\text{'}) \supset \Diamond \tilde{K}_v^{\mathrm{BPP},A}(x \in L),$$

where A is the set of final points of $\mathcal{P} \times V^*$ satisfying 'p running P', then we can effectively modify V^* to obtain V such that (P,V) is an interactive proof system for L.

The protocol V is simply the protocol V^* at the end of which the verifier uses its test for practical knowledge of $x \in L$ to decide whether to accept or reject.

These results tell us that an interactive proof system for L is precisely one that guarantees that if the verifier is running against a good prover, then it will practically know that $x \in L$ at the end of the proof, and it will practically never be fooled (by *any* prover).

5 Zero Knowledge Proof Systems

Informally, an interactive proof system (P,V) is zero knowledge if, whenever $x \in L$, the verifier is able to generate on its own the conversations it could have had with the prover during an interactive proof of $x \in L$. Consequently, the verifier learns nothing as the result of a conversation with the prover (other than the fact that $x \in L$) that it could not have learned on its own by generating the conversation itself.

To make this precise, we first recall the notion of polynomial indistinguishability (cf. [GMR85, GMW86, Ore87]). Suppose we have some domain *Dom* whose elements are of the form $(x,\bar{y})$, where $\bar{y}$ is a vector of values. Further suppose for each $(x,\bar{y}) \in Dom$ we have two random variables $U_{x,\bar{y}}$ and $V_{x,\bar{y}}$ with two associated probability distributions. The families $\{U_{x,\bar{y}} : (x,\bar{y}) \in Dom\}$ and $\{V_{x,\bar{y}} : (x,\bar{y}) \in Dom\}$ are said to be *polynomially indistinguishable* if for every probabilistic, polynomial-time algorithm M and every constant k there exists a constant $N_{M,k}$ such that for all x with $|x| \geq N_{M,k}$ and all $\bar{y}$ with $(x,\bar{y}) \in Dom$ we have

$$|\,Pr[M \text{ accepts } U_{x,\bar{y}}] - Pr[M \text{ accepts } V_{x,\bar{y}}]| \leq |x|^{-k}.$$

It is important to notice that the probability is being taken over both the coin flips of M *and* the distributions of $U_{x,\bar{y}}$ and $V_{x,\bar{y}}$.

Other notions of indistinguishability are defined in [GMR85] (i.e., *perfect indistinguishability*, *statistical indistinguishability*, and *computational indistinguishability*). Since polynomial indistinguishability is implied by each of these notions, our results, which are proven for polynomial indistinguishability, hold for these other notions as well.

Finally, an interactive proof system (P,V) for L is said to be *zero knowledge* (cf. [GMR85, GMW86]) if for every verifier V^* there is a probabilistic Turing machine M_{V^*} such that

1. $M_{V^*}(t,x)$ runs in expected time polynomial in $|x|$, and

2. the families $\{(P(s),V^*(t))(x) : (x,s,t) \in \mathit{Dom}\}$ and $\{M_{V^*}(t,x) : (x,s,t) \in \mathit{Dom}\}$ are polynomially indistinguishable, where $(x,s,t) \in \mathit{Dom}$ iff $x \in L$, s is a possible input for P, and t is a possible input for V^*.

5.1 Knowledge and Zero Knowledge

In this section we formalize the intuition that if the verifier can learn a fact φ at the end of a zero knowledge proof of $x \in L$, then the verifier can deduce φ from $x \in L$ on its own at the beginning of the proof. First, we need a short definition. We say that φ is a fact *about the initial state* (in a system $\mathcal{R}$) if $(r,m) \models \varphi$ iff $(r',m') \models \varphi$ for all points (r,m) and (r',m') of $\mathcal{R}$ with $r(0) = r'(0)$. Thus φ is a fact about the initial state if its truth at a given point in a run depends only on the initial state in that run.

The following theorem captures our intuition that the prover does not leak any information to the verifier during a zero knowledge proof of $x \in L$ other than the fact $x \in L$. Roughly speaking, it says that if $x \in L$ and the verifier has a nontrivial chance of learning φ at the end of a proof of $x \in L$, then the verifier can already deduce φ from $x \in L$ on its own without interacting with the prover. Consequently, provided $x \in L$, the only information that a prover leaks to the verifier in a zero knowledge proof of $x \in L$ are facts that follow from $x \in L$. The proviso that $x \in L$ is crucial here. There is nothing in the definition of a zero knowledge proof to stop the prover from leaking all sorts of information when $x \notin L$.

Theorem 5: Let (P,V) be a zero knowledge proof system for L, let V^* be an arbitrary verifier, and let φ be a fact about the initial state. For every set A of final points in $P \times V^*$ and every k there exist constants c and N such that

$$P \times V^* \models (x \in L \wedge \mathit{init}) \supset K_p^{1-c|x|^{-k}}[\Diamond \tilde{K}_v^{\mathrm{BPP},A}\varphi \supset \tilde{K}_v^{\mathrm{BPP},B}(x \in L \supset \varphi)]$$

where B is the set of initial points in $P \times V^*$ satisfying $x \in L$, $|x| \geq N$, and $Pr(\Diamond K_v^{\mathrm{BPP},A}\varphi) \geq |x|^{-k}$.

Proof: Fix a set A and a constant k. The definition of knowing φ with respect to A ensures the existence of a test M for φ that is sound everywhere and complete on A. The definition of a zero knowledge proof system (P,V) ensures the existence of a Turing machine $M_{V^*}(t,x)$ that approximates $(P(s),V^*(t))(x)$. Informally, the proof proceeds as follows. Suppose that from an initial point $(r,0)$ the probability of reaching a final point at which the test M indicates that φ holds is at least $|x|^{-k}$. Suppose that from this initial point we run $M_{V^*}(t,x)$ to generate a run of $(P(s),V^*(t))(x)$ and apply the test M to its final state. If we repeat this procedure roughly $|x|^k$ times, then with high probability we will generate a run at whose final state the test M will succeed, and hence with high probability we will learn that if $x \in L$ (and hence the simulating Turing machine $M_{V^*}(t,x)$ is accurate), then φ must hold. The details of the proof are left to the full paper. □

We note that the same result holds when we replace practical knowledge by knowledge with respect to a set of points, but as we have seen in Section 4 the notion of practical knowledge seems to be of greater relevance to interactive protocols.

Stepping back and looking at the statement of Theorem 5, we see that the result is slightly unsatisfactory. Notice that in the system $P \times V^*$ the verifier protocol V^* is fixed, and hence known to the prover. The intuition behind zero knowledge proofs, however, is that even though the prover does not know the identity of the verifier, the prover knows that the verifier learns nothing at the end of the proof other than facts that follow from $x \in L$. That is, our intuition suggests that the statement of Theorem 5 should hold in the system $P \times \mathcal{V}^{pp}$. We cannot prove such a result due to the order of quantification in the definition of zero knowledge guaranteeing only that for every verifier V^* there is a Turing machine $M_{V^*}(t,x)$ approximating the distribution of $(P(s),V^*(t))(x)$. The problem is that because the Turing machine M_{V^*} cannot in general be chosen in some uniform way, and because the tests for knowledge we allow must be uniform in V^*, we do not have a test for computing facts at the beginning of all runs in $P \times \mathcal{V}^{pp}$. One solution to our problem is provided by the notion of black-box zero knowledge. An interactive proof system (P,V) for L is said to be *strongly black-box zero knowledge* (cf. [Ore87]) if there is a probabilistic Turing machine M such that

1. $M(V^*,t,x)$ runs in expected time polynomial in $|x|$, and

2. $\{(P(s),V^*(t))(x) : (x,V^*,s,t) \in \mathit{Dom}\}$ and $\{M(V^*,t,x) : (x,V^*,s,t) \in \mathit{Dom}\}$ are polynomially indistinguishable, where $(x,V^*,s,t) \in \mathit{Dom}$ iff $x \in L$, V^* is a possible verifier protocol, s is a possible input for P, and t is a possible input for V^*.

If (P, V) is a strongly black-box zero knowledge proof system for L, then we can prove the analogue of Theorem 5 in the system $P \times \mathcal{V}^{pp}$ instead of $P \times V^*$.

Unfortunately, as the name suggests, the notion of strongly black-box zero knowledge is too strong. The problem is that in practice $M(V^*, t, x)$ runs V^* as a subroutine on input x. Even if M runs V^* only once, the running time of M is at least as great as the running time of V^*. Consequently, even if we restrict our attention to polynomial-time V^* as input to M, since the polynomial bounding the running time of V^* is different for every V^*, the running time of M will not be bounded by a single polynomial. Oren avoids this problem in his definition of black-box zero knowledge by charging only one time step for a call to V^*. Thus, he is essentially viewing M as an oracle machine (rather than a purely polynomial-time Turing machine). We could modify our definitions to allow for knowledge with respect to oracle machines, but a more natural solution is to modify the measure we use of a test's complexity. In particular, suppose we consider tests for facts that run at a point (r, m) in time polynomial in $|x|$, the running time of V^*, and the description of V^*, where r is a run with input x in which the verifier is running the protocol V^*. Then, defining a notion of practical knowledge with respect to such tests, the analogue of Theorem 5 follows with precisely the same proof. We note that all zero knowledge protocols we are aware of satisfy Oren's notion of black-box zero knowledge.

5.2 Generation and Zero Knowledge

In the previous subsection we formalized the notion that the verifier in a zero knowledge proof learns essentially no fact other than what the prover explicitly set out to prove. This is not, however, the strongest notion of security one could hope for. It would also be desirable to show that, as a result of interacting with the prover, the verifier cannot *do* anything that it could not do before the interaction. We abstract the idea of the verifier being able to do something as *knowing how to generate a y such that* $R(x, y)$. For example, if $R(x, y)$ holds precisely when y is a Hamiltonian circuit in a graph x on the input tape, then being able to generate a y such that $R(x, y)$ means being able to find a Hamiltonian circuit in the graph x. Notice that, as in the case of Hamiltonian circuits, most natural relations R are testable in BPP. That is, there is a probabilistic algorithm running in time polynomial in $|x|$, accepting (x, y) with probability at least $2/3$ if $R(x, y)$, and rejecting (x, y) with probability $2/3$ if $\neg R(x, y)$. We restrict our attention to such BPP testable relations here to simplify our exposition.

Just as we have said that the verifier knows a fact φ if it has an algorithm to test for φ, we would like to say that the verifier knows how to generate a y satisfying $R(x, y)$ if it has an algorithm to generate such y. In previous sections we considered tests for facts φ that were sound everywhere and correct on a set A of points. Here, although there are no conditions analogous to soundness and completeness, we consider algorithms that do a "good job" of generating y's such that $R(x, y)$ on a set A of points, but may not perform so well off A. We say that the verifier *knows how to BPP-generate a y satisfying* $R(x, y)$ *with respect to a set* A *of points* in a system $\mathcal{R}$ if there is a probabilistic algorithm, that, at all points (r, m) of A, takes as input the verifier's local state and outputs with probability at least $2/3$ a string y satisfying $R(x, y)$. Formally, we write $(r, m) \models G_v^{\mathrm{BPP},A} y.R(x, y)$ iff

1. $(r, m) \in A$, and
2. there is a probabilistic Turing machine M that at points $(r', m') \in \mathcal{R}$ takes the verifier's local state as input, runs in time polynomial in $|x|$, and, if $(r', m') \in A$, outputs with probability at least $2/3$ a string y satisfying $R(x, y)$.

We have the following analogue to Theorem 5:

Theorem 6: Let (P, V) be a zero knowledge proof system for L, let V^* be an arbitrary verifier, and let $R(x, y)$ be a relation testable in BPP. For every set A of final points in $P \times V^*$ and every k, there exist c and N such that

$$P \times V^* \models (x \in L \wedge \mathit{init}) \supset K_p^{1-c|x|^{-k}} [\Diamond G_v^{\mathrm{BPP},A} y.R(x, y) \supset G_v^{\mathrm{BPP},B} y.R(x, y)]$$

where B is the set of initial points in $P \times V^*$ satisfying $x \in L$, $|x| \geq N$, and $Pr(\Diamond G_v^{\mathrm{BPP},A} y.R(x, y)) \geq |x|^{-k}$.

Intuitively, this theorem says that if the verifier has a nonnegligible chance of being able to generate a y satisfying $R(x, y)$ by talking to the prover, then the verifier can generate such a y on its own. We note that this theorem has a number of natural extensions. One simple extension is from generating y satisfying relations $R(x, y)$ to generating y satisfying facts $\varphi(y)$ about the verifier's entire initial state. Another simple extension is, along the lines of practical knowledge, a notion of knowing how to generate, denoted by $\tilde{G}_q^{\mathrm{BPP},A} y.R(x, y)$, where the algorithm may on a small fraction of the set A fail to generate y such that $R(x, y)$. A final extension, using black-box zero knowledge, allows us to prove an analogous result in the system $P \times \mathcal{V}^{pp}$. The details are left to the full paper.

The ability to test the relation R in BPP is crucial to the proof of Theorem 6. Recall that in the proof of Theorem 5 the verifier tests for the fact ψ by repeatedly generating runs and testing for φ at the end of each run. Since this test for φ is sound, the verifier can accept as soon as this test for φ accepts. Here, however, since there is no notion analogous to soundness, the verifier has no way of knowing which of the many y's it generates satisfies $R(x,y)$ and should be output unless the relation $R(x,y)$ can be tested in BPP. We discuss analogues of Theorem 6 when $R(x,y)$ is not testable in BPP in the full paper.

6 Resource-bounded provers

In an interactive proof system as defined in [GMR85], the prover is assumed to be infinitely powerful. In practice, however, a prover is not infinitely powerful and may have no more computational power than the verifier. Fortunately, a probabilistic, polynomial-time prover with some "secret information" on its work tape is able to carry out many of the interesting interactive protocols. In the case of the graph isomorphism protocol given in [GMW86], for example, this secret information is an isomorphism between the graphs on the input tape. Since the context of such weak (polynomial-time) provers is actually the context of most practical interest, the type of security afforded by zero knowledge protocols in this context is an important question, and the subject of our final section.

In order to study zero knowledge proofs in this context, we define the notion of a weak interactive proof system, a direct modification of the definition of an interactive proof system for L. We define a *weak interactive protocol* to be an interactive protocol (P,V) where both P and V run in probabilistic, polynomial-time. We define a *weak interactive proof system* (P,V) *for a language* L just as we defined an interactive proof system for L except that we require (P,V) to be a weak interactive protocol and we restrict the quantification of P^* in the soundness condition to be only over probabilistic, polynomial-time machines, rather than over all machines. As the following lemma shows, however, weak interactive proofs of language membership are not very interesting.

Lemma 7: There is a weak interactive proof system for L iff L is in BPP.

Thus, an interesting weak interactive proof cannot be simply a proof of language membership; it must reveal something about the prover's local state, and hence, since the prover's knowledge is determined by its local state, it must reveal something about the prover's knowledge. Consider the zero knowledge proofs of graph isomorphism and three-colorability given in [GMW86]. These proofs can be carried out by a weak prover with the appropriate information on its worktape. And in both cases, the verifier obtains information about the prover's knowledge as well as about language membership. In the case of graph isomorphism, the verifier learns that with high probability, the prover can generate an isomorphism between the graphs in question. Similarly, in the case of three-colorability, the verifier learns that with high probability the prover can generate a three coloring of the graph in question. It is well-known (see [HM84, MDH86]) that information about the prover's knowledge can dramatically affect the verifier's knowledge about the world. For example, in the case of three-colorability, information about the prover's knowledge may indicate to the verifier that the prover has with high probability communicated with the entity that generated the three-colorable graph.

In order to study proofs of the prover's knowledge, we extend the definition of a weak interactive proof of language membership to that of a weak interactive proof about the prover's initial state, where a fact $R(P^*,x,s)$ is about the prover's initial state if its truth depends only on the prover's protocol P^*, its initial work tape s, and the common input x. The definition of a weak interactive proof of $R(P^*,x,s)$ is obtained simply by replacing all occurrences of $x \in L$ by $R(P^*,x,s)$ in the definition of a weak interactive proof of language membership. Formally, we define a *weak interactive proof system* for a fact R about the prover's initial state to be a weak interactive protocol (P,V) such that

- *Completeness*: For every k and sufficiently large x, and for every s and t, if $R(P,x,s)$ then
$$Pr\left[(P(s),V(t))(x) \text{ accepts}\right] \geq 1-|x|^{-k}.$$
- *Soundness*: For every k and sufficiently large x, for every probabilistic, polynomial-time P^*, and for every s and t, if $\neg R(P^*,x,s)$ then
$$Pr\left[(P^*(s),V(t))(x) \text{ accepts}\right] \leq |x|^{-k}.$$

The reader may wonder why we consider weak interactive proofs of facts about the prover's initial state that depend on the prover's protocol as well as its worktape. Suppose $R(x,s)$ is a fact about the prover's worktape and the common input; that is, the truth of $R(x,s)$ depends only on the prover's worktape s and the common input x (and not on the prover's protocol). Let us define $dom(R)$ to be the set $\{x : R(x, s) \text{ for some } s\}$.

Lemma 8: A weak interactive protocol (P, V) is a weak interactive proof system for a fact R about the prover's worktape and the common input iff

1. for all sufficiently large x and for all s, we have $R(x, s)$ iff $x \in dom(R)$; and
2. $dom(R)$ is in BPP.

This lemma says that if there is a weak interactive proof of a fact R about the prover's worktape and the common input, then R is essentially uninteresting. In particular, with the exception of a few small values of x, $R(x, s)$ holds for some s iff $R(x, s')$ for all s'. Consequently, R is essentially determined by $dom(R)$. Since $dom(R)$ is in BPP, the prover can determine whether R holds (for sufficiently large x) without even interacting with the prover. Consequently, a fact R about the prover's initial state having only nontrivial weak interactive proofs must necessarily be a fact depending on the prover's protocol, and hence on the prover's entire initial state. Since the prover's knowledge is determined by its local state, such a weak interactive proof may be viewed as a proof of the prover's knowledge. In fact, we note that even in the context of infinitely powerful provers an interactive proof of $x \in L$ is not just a proof of $x \in L$ but a proof the prover knows $x \in L$. The fact that all interesting interactive proofs must be proofs of the prover's knowledge is obscured in the context of infinitely powerful provers since $x \in L$ holds iff the prover knows $x \in L$. In the context of weak prover, however, these facts are not equivalent.

We have defined a natural notion of interactive proof in the context of weak provers, and we have shown that the only nontrivial interactive proofs in this context are proofs about the prover's knowledge. While our definition is a direct modification of the definition in the case of strong provers, it is not initially clear that our definition is the most appropriate in the context of weak provers, and hence that our results are more than simply artifacts of our definition. As evidence supporting our definition, we now show that, under certain natural conditions, both interactive proof systems [FFS87, TW87] involving weak provers that have appeared in the literature are instances of weak interactive proofs. Not surprisingly, in light of our previous results, these proof systems concern proofs of the prover's knowledge.

We focus here on [TW87], and leave the discussion of [FFS87] to the full paper. In [TW87] we find the following definition (modified slightly for the sake of consistency with the rest of this abstract). Given a binary relation R, a weak interactive protocol (P, V) is said to be *an interactive proof that the prover can generate some y satisfying* $R(x, y)$ if the following conditions are satisfied:

- *Completeness*: For every k and sufficiently large x, and for every s and t, if $R(x, s)$, then

$$Pr\left[(P(s), V(t))(x) \text{ accepts}\right] \geq 1 - |x|^{-k}.$$

- *Soundness*: For every probabilistic, polynomial-time P^* there is a probabilistic Turing machine M_{P^*} running in time polynomial in $|x|$ such that for all k and sufficiently large x, and for all s and t,

$$Pr\left[V \text{ accepts at } (r, m) \supset R(x, M_{P^*}(r_p(m)))\right] \geq 1 - |x|^{-k}$$

where the probability is taken over the runs of $(P^*(s), V(t))(x)$ and the coin flips of M_{P^*}.[5]

While we would like to show that every interactive proof that the prover can generate some y satisfying $R(x, y)$ is a weak interactive proof, this is not quite true. To see this, notice that the definition of a weak interactive proof requires that $(P(s), V(t))(x)$ accept with probability very close to 0 or 1, while an interactive proof of [TW87] allows $(P(s), V(t))(x)$ to accept with any probability as long as P is able to generate a y satisfying $R(x, y)$. We can prove, however, that the following is a necessary and sufficient condition for an interactive proof of [TW87] to be a weak interactive proof:

- *Correctness*: For every k and sufficiently large x, and for every s and t, if $R(x, s)$ does not hold, then $Pr\left[(P(s), V(t))(x) \text{ accepts}\right] \leq |x|^{-k}$.

In other words, the good prover succeeds in convincing the good verifier to accept only when $R(x, s)$ holds. We note that the correctness condition can be satisfied iff $R(x, y)$ is testable in BPP. Since this seems to be the most relevant context in practice, this seems to be a natural restriction. We have seen that the correctness condition is necessary for an interactive proof

[5] We note that the soundness condition in [TW87] actually quantifies over *all* Turing machines P^* and not just over polynomial-time P^*. Since, however, the motivation for considering weak provers is that in practice all agents are restricted to polynomial-time, our restriction does not seem unnatural. Furthermore, we note that the machine M_{P^*} is allowed in [TW87] to run in expected polynomial-time. In the context of BPP-testable relations R, however, the context we find of most interest, we can assume without loss of generality that the machine M_{P^*} runs in polynomial-time.

of [TW87] to be a weak interactive proof. To see that this condition is sufficient, let $R'(P^*, x, s)$ be the fact "$(p$ *running* $P \wedge R(x,s)) \vee (p$ *running* $P^* \neq P \wedge$ *the soundness condition holds for* P^**)*." (Note that R' depends on the prover's protocol as well as the work tape, and is a fact about the prover's initial state.) We now have the following.

Proposition 9: (P, V) is an interactive proof satisfying the correctness condition that the prover can generate a y such that $R(x, y)$ iff (P, V) is a weak interactive proof system for R'.

We can show, in addition, that the proof systems of [FFS87] satisfying the correctness condition above are also instances of a weak interactive proof system.

Having shown that, in light of Proposition 9, our definition of a weak interactive proof system seems to be an appropriate definition, let us turn to the study of the security afforded by such protocols. Our definition of a weak interactive proof is a direct modification of the definition of an interactive proof of language membership. We can also directly modify the definition of a zero knowledge proof of language membership to obtain a definition of a zero knowledge weak interactive proof. Not surprisingly, analogues of all our previous results for interactive proofs hold in the case of weak interactive proofs, with essentially the same proofs. Rather than restating all the results here, we focus on one of them, the analogue of Proposition 1. If R is a fact about the prover's initial state, then we say $(r, m) \models R$ if $R(P^*, x, s)$, where P^* is the protocol that p is running in r, x is the common input in the initial state $r(0)$, and s is the contents of p's work tape in $r(0)$.

Proposition 10: A weak interactive protocol (P, V) is a weak interactive proof system for a fact R about the prover's initial state iff the following conditions are satisfied:

- *Completeness*: For every k there exists c such that

$$P \times V \models init \supset Pr[R \supset \Diamond accept] \geq 1 - c\,|x|^{-k}$$

- *Soundness*: For every k there exists c such that

$$\mathcal{P}^{pp} \times V \models init \supset Pr[\Diamond accept \supset R] \geq 1 - c\,|x|^{-k}.$$

Thus, we have replaced the occurrences of $x \in L$ in Proposition 1 by R, and used $\mathcal{P}^{pp}$ rather than $\mathcal{P}$ in the soundness condition since we are restricting to weak provers. As we mentioned in our discussion in Section 3, while the order of quantification in the statement of soundness is irrelevant in the case of strong provers, it does play a role in the case of weak provers. In particular, if we had stated our soundness condition so that the choice of "sufficiently large x" might depend on the protocol P^*, all we would be able to prove is that for every k and every protocol P^*, there exists c such that

$$P^* \times V \models init \supset Pr[\Diamond accept \supset R] \geq 1 - c\,|x|^{-k}.$$

We remark that the weak interactive protocols resulting from the interactive proofs and zero knowledge proofs we are aware of satisfy the stronger notion of soundness we have used in our definition.

In addition to proving the analogues of results holding in the context of strong provers, we can reason about the interactive proofs of [FFS87, TW87] directly in terms of the notions of knowledge and generation we have defined in previous sections. For example, we can characterize proofs that the prover can generate some y satisfying $R(x, y)$ just as we characterized interactive proofs, in the case that $R(x, y)$ is testable in BPP.

Proposition 11: Given a relation $R(x, y)$ testable in BPP, a weak interactive protocol (P, V) is a weak interactive proof that the prover can generate some y satisfying $R(x, y)$ iff the following conditions are satisfied:

- *Completeness*: For every k there exists c such that

$$P \times V \models init \supset Pr[R(x,s) \supset \Diamond accept] \geq 1 - c\,|x|^{-k}$$

- *Soundness*: For every probabilistic, polynomial-time P^*,

$$P^* \times V \models accept \supset \tilde{G}_p^{\mathrm{BPP},A} y.R(x,y)$$

  where A is the set of points of $P^* \times V$ at which the verifier has accepted.

Notice that in the soundness condition, we have $accept \supset G_p^{\mathrm{BPP},A} y.R(x,y)$ rather than $\Diamond accept \supset G_p^{\mathrm{BPP},A} y.R(x,y)$. The first clause says that the prover can generate some y such that $R(x, y)$ at the point when the verifier accepts, as required by [TW87], and not at the initial point as would be the case with the second clause. This is one of the differences between the definitions of [TW87] and [FFS87]. A second difference between the two definitions is that the soundness condition of [FFS87] is such that we can state the soundness condition above in terms of the system $\mathcal{P}^{pp} \times \mathcal{V}$ instead of $P^* \times V$. We return to these points in the full paper.

7 Conclusions

The main contribution of this work lies in suggesting notions of knowledge appropriate for interactive proofs, characterizing interactive proofs in terms of these notions, and proving, again in terms of these notions, that the prover in a zero knowledge proof system does not leak any information other than the fact it set out to prove. Roughly speaking, we have shown that a zero knowledge proof system for $x \in L$ satisfies the following property, which we call *knowledge security*: the prover is guaranteed that, with high probability, if the verifier will practically know a fact φ at the end of the proof, it practically knows $x \in L \supset \varphi$ at the start. We have also formalized the notion of knowing how to generate, and shown that zero knowledge proofs also satisfy an analogous property of *generation security*. (The precise formulations of knowledge and generation security are provided by the statements of Theorems 5 and 6.) It is currently an open question whether either of these notions of security characterizes zero knowledge (that is, say, whether an interactive proof that satisfies the property of knowledge security is also a zero knowledge proof). We can show, however, that any protocol that satisfies the knowledge security property is *recognition zero knowledge*, as defined in [DS88]. We discuss this issue in greater detail in the full paper.

We feel that these security results shed some light on the type of security zero knowledge proofs provide. Our theorems provide support for the definitions of interactive proofs and zero knowledge and our model provides a good semantic setting for such an analysis. Some of the definitions, chiefly that of practical knowledge, are quite subtle. Many straightforward definitions one may try fail by being inappropriate for the cryptographic setting and not providing a useful sense in which zero knowledge proof systems provide security. As Feige, Fiat, and Shamir write in [FFS87], "the notion of 'knowledge' is very fuzzy, and a-priori it is not clear what proofs of knowledge actually prove." We hope to have established a framework within which such questions can now be answered.

Acknowledgments

We would like to thank Ron Fagin, Alan Fekete, Oded Goldreich, Moshe Vardi, and Jennifer Welch for their useful comments on these ideas and their suggestions on how to improve the presentation of this work.

References

[BC86] G. Brassard and C. Crepeau, Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond, *Proc. 27th IEEE Symp. on Foundations of Computer Science*, 1986, pp. 188–195.

[CM86] K. M. Chandy and J. Misra, How processes learn, *Distributed Computing* **1**:1, 1986, pp. 40–52.

[DM86] C. Dwork and Y. Moses, Knowledge and common knowledge in a Byzantine environment I: crash failures (extended abstract), *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference* (J. Y. Halpern, ed.), Morgan Kaufmann, 1986, pp. 149–170. To appear in *Information and Computation.*

[DS88] C. Dwork and L. Stockmeyer, Interactive proof systems with finite state verifiers (extended abstract), 1988. Unpublished manuscript.

[FFS87] U. Feige, A. Fiat, and A. Shamir, Zero knowledge proofs of identity, *Proc. 19th ACM Symp. on Theory of Computing*, 1987, pp. 210–217.

[FH88] R. Fagin and J. Y. Halpern, Reasoning about knowledge and probability: preliminary report, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge* (M. Y. Vardi, ed.), Morgan Kaufmann, 1988, pp. 277–293.

[FI86] M. J. Fischer and N. Immerman, Foundations of knowledge for distributed systems, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference* (J. Y. Halpern, ed.), Morgan Kaufmann, 1986, pp. 171–186.

[For87] L. Fortnow, The complexity of perfect zero knowledge, *Proc. 19th ACM Symp. on Theory of Computing*, 1987, pp. 204–209.

[FZ87] M. J. Fischer and L. D. Zuck, *Relative Knowledge and Belief (Extended Abstract)*, Technical Report YALEU/DCS/TR-589, Yale University, December 1987.

[FZ88] M. J. Fischer and L. D. Zuck, *Uncertain Knowledge in Distributed Systems*, Technical Report YALEU/DCS/TR-604, Yale University, January 1988.

[GHY85] A. Galil, S. Haber, and M. Yung, A private interactive test of a boolean predicate and minimum-knowledge public-key cryptosystems, *Proc. 26th IEEE Symp. on Foundations of Computer Science*, 1985, pp. 360–371.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof systems, *Proc. 17th ACM Symp. on Theory of Computing*, 1985, pp. 291–304.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson, Proofs that yield nothing but their validity and a methodology of crypotgraphic design, *Proc. 27th IEEE Symp. on Foundations of Computer Science*, 1986.

[GS87] S. Goldwasser and M. Sipser, Private coins versus public coins in interactive proof systems, *Randomness and Computation* (S. Micali, ed.), JAI Press, 1987. Extended Abstract available in *Proc. 18th ACM Symp. on Theory of Computing*, 1986.

[Had87] V. Hadzilacos, A knowledge-theoretic analysis of atomic commitment protocols, *Proc. 6th ACM Symp. on Principles of Database Systems*, 1987.

[Hal87] J. Y. Halpern, Using reasoning about knowledge to analyze distributed systems, *Annual Review of Computer Science, Vol. 2*, Annual Reviews Inc., 1987, pp. 37–68.

[HF85] J. Y. Halpern and R. Fagin, A formal model of knowledge, action, and communication in distributed systems: preliminary report, *Proc. 4th ACM Symp. on Principles of Distributed Computing*, 1985, pp. 224–236.

[HM84] J. Y. Halpern and Y. Moses, Knowledge and common knowledge in a distributed environment, *Proc. 3rd ACM Symp. on Principles of Distributed Computing*, 1984, pp. 50–61. A revised version appears as *IBM Research Report RJ 4421*, Aug., 1987.

[HZ87] J. Y. Halpern and L. D. Zuck, A little knowledge goes a long way: simple knowledge-based derivations and correctness proofs for a family of protocols, *Proc. 6th ACM Symp. on Principles of Distributed Computing*, 1987, pp. 269–280. A revised version appears as *IBM Research Report RJ 5857*, Oct., 1987.

[LR86] R. Ladner and J. Reif, The logic of distributed protocols (preliminary report), *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the 1986 Conference* (J. Y. Halpern, ed.), Morgan Kaufmann, 1986, pp. 207–222.

[MDH86] Y. Moses, D. Dolev, and J. Y. Halpern, Cheating husbands and other stories: a case study of knowledge, action, and communication, *Distributed Computing* **1**:3, 1986, pp. 167–176.

[Moo85] R. C. Moore, A formal theory of knowledge and action, *Formal Theories of the Commonsense World* (J. Hobbs and R. C. Moore, eds.), Ablex Publishing Corp., 1985.

[Mos88] Y. Moses, Resource-bounded knowledge, *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge* (M. Y. Vardi, ed.), Morgan Kaufmann, 1988, pp. 261–295.

[MT88] Y. Moses and M. Tuttle, Programming simultaneous actions using common knowledge, *Algorithmica* **3**, 1988, pp. 121–169. A preliminary version appeared in *Proc. 27th IEEE Symp. on Foundations of Computer Science*, 1986.

[NT87] G. Neiger and S. Toueg, Substituting for real time and common knowledge in asynchronous distributed systems, *Proc. 6th ACM Symp. on Principles of Distributed Computing*, 1987, pp. 281–293.

[Ore87] Y. Oren, On the cunning power of cheating verifiers: some observations about zero knowledge proofs, *Proc. 28th IEEE Symp. on Foundations of Computer Science*, 1987, pp. 462–471.

[PR85] R. Parikh and R. Ramanujam, Distributed processes and the logic of knowledge, *Proc. of the Workshop on Logics of Programs*, 1985, pp. 256–268.

[TW87] M. Tompa and H. Woll, Random self-reducibility and zero knowledge interactive proofs of possession of information, *Proc. 28th IEEE Symp. on Foundations of Computer Science*, 1987.