BRICS RS-97-50 Damgård & Pfitzmann: Sequential Iteration of Interactive Arguments

BRICS

Basic Research in Computer Science

Sequential Iteration of Interactive Arguments and an Efficient Zero-Knowledge Argument for NP

Ivan B. Damgård
Birgit Pfitzmann

BRICS Report Series RS-97-50
ISSN 0909-0878 December 1997

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/97/50/`

Sequential Iteration of Interactive Arguments and an Efficient Zero-Knowledge Argument for NP

Ivan Damgård*and Birgit Pfitzmann†

Abstract

We study the behavior of interactive arguments under sequential iteration, in particular how this affects the error probability. This problem turns out to be more complex than one might expect from the fact that for interactive proofs, the error trivially decreases exponentially in the number of iterations.

In particular, we study the typical efficient case where the iterated protocol is based on a single instance of a computational problem. This is not a special case of independent iterations of an entire protocol, and real exponential decrease of the error cannot be expected, but nevertheless, for practical applications, one needs concrete relations between the complexity and error probability of the underlying problem and that of the iterated protocol. We show how this problem can be formalized and solved using the theory of proofs of knowledge.

We also prove that in the non-uniform model of complexity the error probability of independent iterations of an argument does indeed decrease exponentially – to our knowledge this is the first result about a strictly exponentially small error probability in a computational cryptographic security property.

As an illustration of our first result, we present a very efficient zero-knowledge argument for circuit satisfiability, and thus for any NP problem, based on any collision-intractable hash function. Our theory applies to show the soundness of this protocol. Using an efficient hash function such as SHA-1, the protocol can handle about 20000 binary gates per second at an error level of 2^{-50}.

Keywords — Interactive proofs, arguments, proofs of knowledge, computational security, efficient general primitives, multi-bit commitment, statistical zero-knowledge.

1 Introduction

1.1 Background

An interactive argument, also sometimes called a computationally convincing proof system, is a protocol in which a polynomial-time bounded prover tries to

*Aarhus University, BRICS (Basic Research in Computer Science, center of the Danish National Research Foundation

†University of Saarbrücken

convince a verifier that a given statement is true, typically a statement of the form $x \in L$ for a given word x and a language L. Interactive arguments were introduced in various conference papers, which were finally merged into [1].

Compared to the interactive proof systems of [13], arguments require only that polynomial-time provers cannot cheat with significant probability, whereas interactive proofs guarantee this for all provers. On the other hand they enjoy some advantages: Under reasonable computational assumptions, *perfect* zero-knowledge arguments can be constructed for any NP-language [1], i.e., the zero-knowledge property holds not only against polynomial-time verifiers; this is (probably) not possible for proof systems [10]. If one has to choose in practice which of the two properties, soundness of the proof and zero-knowledge, one prefers to have perfect, one should also consider that in order to cheat in an argument, the prover must break the underlying computational assumption *before* the protocol ends, whereas the zero-knowledge property of a computational zero-knowledge proof system breaks down if the computational assumption is broken at any later time.

Traditionally, e.g., in [1] (but see later for some exceptions), the notion of arguments seems to have been to modify the definition of proof systems from [13] by no longer allowing cheating provers unlimited computing resources, and requiring that the success probability of cheating provers is *negligible*, i.e., decreases asymptotically faster than the inverse of any polynomial. We will mention below that even this is not completely trivial to formalize, but take it as a starting point for the moment.

Many natural constructions of interactive arguments and proofs start from an atomic step, in which the prover can cheat with at most some (large) probability ϵ, e.g., $1/2$, under some computational assumption. This step is then iterated to reduce the error probability.

For interactive proof systems, it is easy to see that the error probability of m such sequential iterations is ϵ^m. The same is true for parallel iterations, although this is somewhat less trivial to see [5].

For arguments, the problem is more complex. Bellare et al. show in [6] that parallel iteration for some types of what they call computationally sound protocols fails completely to reduce the error. On the positive side, they also show that parallel iteration of arguments with at most 3 rounds does reduce the error in the way one might expect: it goes down exponentially until it becomes negligible.

1.2 Our Work

The research reported in this paper started by the observation that from the results of [8, 9, 1, 16], we could construct an extremely efficient statistical zero-knowledge argument for Boolean Circuit satisfiability (see section 7). The soundness of the protocol can be based on any family of collision-intractable hash functions. Its asymptotic communication complexity is as good as the best known SAT protocols

[7, 14]; moreover our intractability assumption is weaker. But most importantly, our protocol has the potential of being computationally much more efficient. In practice, if we base the protocol on the well-known hash function SHA-1 [17] and go for an error probability of 2^{-50}, the protocol can process 20000 binary gates per second on a standard PC. To the best of our knowledge, this makes it the computationally most efficient SAT protocol proposed.

The design of the protocol is based on sequential iteration of a basic step, and soundness of the entire protocol seemed intuitively to follow trivially from the collision-intractability of the hash function used. However, the real proof of soundness of the protocol turned out to be much more complicated than we expected. It became apparent that we were in fact looking at a general problem, namely how to treat sequential iteration of arguments properly. We believe that there are several reasons for studying this scenario in some detail:

- The exact exponential decrease of the error probability for independent sequential iterations does extend from interactive proofs to arguments, but only if we allow cheating provers to be non-uniform. This is not hard to prove, but interesting because we are not aware of any previous proof that a computational security property has an exponentially (in contrast to just negligibly) small error probability (under the usual cryptologic assumptions). In the uniform case, things are more complicated, as detailed below. No general sequential iteration lemma for arguments seems to have appeared in the literature. protocols.

- Most iterated arguments appearing in the literature (including [1] and the protocol presented in this paper) in fact do not use *independent* iterations; instead the soundness of all iterations is based on a single instance of some computational problem, chosen for the entire protocol. In practice this is typically a lot more efficient than having to choose a new instance for every execution, and may in fact be the only option: if one wants to base a protocol on standard hash functions such as SHA-1, there is only one instance to work with. Note that this scenario is not a special case of the case with independent iterations. It requires separate treatment and we have found that in fact it allows more precise results to be shown.

- For a practical application, it is not enough to have a result of the form: "if there exists a polynomial-time prover that cheats m iterations with probability δ, then there is a polynomial-time prover that cheats one iteration with probability ϵ". The basis for security of one iteration will typically be a concrete hard problem instance, such as a collision-intractable hash function or a large integer that should be hard to factor. The protocol designer is more likely to ask "If I'm willing to assume that this particular hash function cannot be broken in time T with probability ϵ, and I'm happy with error probability δ, how much time would an enemy need to invest in breaking my protocol?" For this, the concrete complexity of reductions involved in proofs is important. We will focus on this in the following.

2 Independent Iterations in the Non-Uniform Model

Our model of uniform feasible computation is probabilistic polynomial-time interactive Turing machines as defined in [13]. For non-uniform computations, we use probabilistic polynomial-time interactive Turing machines with polynomial advice. This means that the machine obtains an additional "advice" input which is a function of the length n of the normal input, and whose length is bounded by a polynomial in n. [1]

For the non-uniform case, we use the following precise definition of arguments:

Definition 2.1 Non-uniform definition of interactive arguments
Let L be a language, (P, V) a pair of probabilistic polynomial-time interactive algorithms with a common input x, and $\epsilon : \{0,1\}^* \to [0,1]$ a function. We say that (P, V) is an interactive argument for L in the non-uniform model with soundness error ϵ *if it has the following properties:*

- *Completeness: If $x \in L$ and P is given an appropriate auxiliary input (depending on x), the probability that V rejects is negligible in $|x|$.*
- *Soundness: For any non-uniform probabilistic polynomial-time algorithm P^* (a cheating prover), there is at most a finite number of values $x \notin L$ such that V accepts in interaction with P^* on input x with probability larger than $\epsilon(x)$.*

To model more precisely how the systems would be used in practice, one can refine this definition by letting the level of security be determined by special security parameters, instead of the length of the input. One would typically use two such parameters, one for computational aspects and one for the tolerated error probability. This was omitted in this extended abstract for brevity and for similarity with other definitions.

We now formally define sequential composition and a view of them as game trees that will be used in most of the proofs.

Definition 2.2 (Iteration and proof trees) *If (P, V) is an interactive argument and $m : \mathbb{N} \to \mathbb{N}$ a polynomial-time computable function, the m-fold iteration or sequential composition is the pair of algorithms (P^m, V^m) which, on input x, execute P or V, respectively, $m(|x|)$ times sequentially and independently. V^m accepts if all its executions of V accept. Typically, m is a constant or the identity function*

[1] It is well-known that this is equivalent to using circuit complexity, the more usual model of non-uniformity in cryptology. In particular, a polynomial-size circuit family can be simulated by a polynomial-time Turing machine that takes the description of the circuit for the given input length n as advice input.

(i.e., the number of iterations equals the input length, which serves as the security parameter).

Cheating provers will be denoted by P^{m*}; this notation only designates that this is a non-uniform probabilistic polymial-time algorithm that can interact with V^m.

The proof tree *for V^m, such a P^{m*}, and a fixed common input x has a node for each state that P^{m*} may be in just before an iteration of V. A node has an outgoing edge for each possible set of random choices that P^{m*} and V might make in the following iteration; if V rejects in the iteration with this set of random choices, we truncate off the corresponding edge.*

For this non-uniform case, we can show the following theorem about strictly exponential decrease of the error probability.

Theorem 2.3 *Let (P, V) be an interactive argument for a language L in the non-uniform model with soundness error ϵ, where ϵ is a constant, and m a polynomial-time computable function. Then (P^m, V^m) is an interactive argument for L with soundness error ϵ^m (where ϵ^m is the function that maps x to $\epsilon(x)^{m(|x|)}$).*

Proof Assume, for contradiction, that there were a non-uniform probabilistic polynomial-time prover P^{m*} and an infinite number of values $x \notin L$ such that V^m accepts in interaction with P^{m*} on input x with probability larger than $\epsilon(x)^{m(|x|)}$. This implies that there is an infinite number of different input lengths n such that a value x_n with this property and of length n exists.

For each value x_n, consider the corresponding proof tree (for the given P^{m*}), and abbreviate $\epsilon(x_n)$ by ϵ and $m(|x_n|)$ by m. If T is the number of possible sets of random choices in each iteration, success probability for P^{m*} larger than ϵ^m means that the tree has at least $T^m\epsilon^m$ leaves. It easily follows that there is a node with at least $T\epsilon$ children. We fix one such a node N_n, say the first one in a standard enumeration of the nodes of the tree.

Now we exploit that we allow non-uniform provers: The particular strategy corresponding to the node with large success probability can be given as advice. More precisely, we construct a prover P^* that cheats one iteration as follows: It takes its advice as a starting state for running P^{m*} for as long as needed to interact with V. The value $f(n)$ of the advice function for each input length n is the state that labels the node N_n. This P^* is obviously probabilistic polynomial-time and successfully cheats V on the infinite set of values x_n with probability larger than ϵ, in contradiction to the security of the given argument. □

As to the concrete complexity of the reduction, note that the running time of P^* is bounded by that of P^{m*} on the same value x. (We have not shown that P^* can cheat with a too large probability on equally many values $x \notin L$ as P^{m*}. But we have shown that if x is long enough so that P^* cannot cheat with a too large probability on any value of this length, then the same holds for P^{m*}.)

3 Uniform Definitions

Before presenting a specialized treatment of the notion of arguments based on a fixed instance of a computational problem, let us review general definitions of arguments in the uniform model. The specialized definition will be a refinement of such a definition.

A natural version can be derived from the non-uniform definition by simply restricting cheating provers to uniform computations. We assume that this is what the authors of [1] had in mind, and it comprises the formal definitions specifically for negligible and constant error in [11], and the language-recognition case of [6]. We will show below, however, why we did not call it a uniform definition.

Definition 3.1 Semi-uniform definition of interactive arguments
We say that (P, V) is an interactive argument for L in the semi-uniform model with soundness error ϵ *under exactly the same conditions as in 2.1, except that the word "non-uniform" in the soundness is replaced by "uniform".*

$|x|$. *We say that (P, V) is an interactive argument for L in the semi-uniform model with* negligible *soundness error if this property holds for any function $\epsilon(x) = |x|^{-c}$.*

The reason why we did not simply call this definition uniform is the treatment of the common input x. Typically, the goal with a uniform definition is to be able to prove that a protocol is secure according to the definition under the assumption that a certain computational problem is hard for *uniform* algorithms. If one is willing to make a stronger, non-uniform complexity assumption, it is more useful to also prove a stronger, non-uniform security property for the protocol.

However, for Definition 3.1, one will usually need a non-uniform complexity assumption.

Intuitively, one can see this from trying the usual reduction proof: If there exists a prover P^* contradicting Definition 3.1, there exists an infinite set K of inputs on which P^* can cheat V. In order to exploit P^* in a reduction to an algorithm breaking the assumption, one would for each input length need to get hold of an element from K, if it exists. Nothing in the scenario guarantees us that this can be done efficiently; thus the only solution seems to be to give the elements of K as advice. One can actually construct realistic counterexamples where the existence of a non-uniform algorithm F breaking the assumption can be exploited in a uniform prover P^* contradicting Definition 3.1: In proof systems like [1] that can be broken if one or more instances of the underlying problem are broken, P^* can try to use its input x as advice in calls to F to break these instances. On an infinite set of inputs x this will work (at least if x may be longer than the underlying instances by a polynomial factor, which should usually be no problem).

Thus we also propose a fully uniform definition. It contains a polynomial message finder like definitions of the secrecy in encryption schemes [12]. We can leave completeness unchanged for our purposes; for many practical purposes one would additionally require the existence of a probabilistic polynomial-time algorithm that generates instances together with the appropriate auxiliary input.

Definition 3.2 Fully uniform definition of interactive arguments
Let L be a language, (P,V) a pair of probabilistic polynomial-time interactive algorithms with a common input x, and $\epsilon : \mathbb{N} \to [0,1]$ a function. We say that (P,V) is an interactive argument for L in the fully uniform model with soundness error ϵ *if it has the following properties:*

- *Completeness: If $x \in L$ and P is given an appropriate auxiliary input (depending on x), the probability that V rejects is negligible in $|x|$.*
- *Soundness: A cheating prover is modeled by two (uniform) probabilistic polynomial time algorithms M^* and P^*, called message finder and main prover. The input for M^* is a security parameter k, and its output should be a value $x \notin L$ of length k and a value $view_{M^*}$, which serves as local knowledge that the cheating prover stores between the two steps. We consider the joint probability distribution if first M^* is run on input k and outputs two values $(x, view_{M^*})$, and then V on input x interacts with P^* on input $(x, view_{M^*})$.*

  The soundness requirement is that for any such pair (M^*, P^*), there exists at most a finite number of integers k such that the probability (in the distribution just defined) that $|x| = k$, $x \notin L$, and that V accepts is larger than $\epsilon(k)$.

Negligible soundness error is defined as above.

It is clear that any interactive argument for L in the semi-uniform model with soundness error ϵ is also an interactive argument for L in the fully uniform model with at most the same soundness error.

The definition and results in the following sections are all phrased in the fully uniform model. The results and proofs carry over easily to non-uniform provers, using essentially the same reductions.

4 Definition of Fixed-Instance Arguments

We now consider the case where the soundness of the entire iterated protocol is based on a single instance of a computational problem, which is chosen by the verifier initially. This will almost always be the scenario in a practical application, also all protocols in [1] are of this form. In this case, it is clear that the error probability of the iterated protocol cannot decrease strictly exponentially in the number of iterations, even in the non-uniform model, because it always suffices to break the one given instance.

The intuition behind the term "based on an instance of a problem" is that if any prover can convince the verifier on input $x \notin L$ with probability greater than some ϵ, then the prover can solve the problem instance. We call such a definition *relative* (to the hardness of the underlying problem) and the definitions presented so far *absolute* in comparison. To capture what it means that the prover "can solve" the problem instance, we use the theory of proofs of knowledge. We do not claim that the resulting definition covers any conceivable argument where some part is iterated while another is kept constant, but it does cover all known examples of what people have understood by basing an iteration on one problem instance.

Before we give our new relative definition of soundness of an interactive argument, we briefly recall the definition from [4] of a proof of knowledge, with minor modifications to match our context. For any binary relation R, let $R(z)$ be the set of y's such that $(z, y) \in R$, and $L_R = \{z |\ R(z) \neq \emptyset\}$.

Definition 4.1 (Proof of knowledge) *Let R be a binary relation, and $\kappa : \{0,1\}^* \rightarrow [0,1]$. Let V be a probabilistic polynomial-time interactive Turing machine. We say that V is a* knowledge verifier for R with knowledge error κ *if the following two conditions hold:*

- *Non-triviality (completeness): There is a prover P such that V always accepts in interaction with P for all inputs $z \in L_R$.*
- *Validity (soundness): There is a probabilistic oracle machine K (the universal knowledge extractor) and a constant c such that for every prover P^* and every $z \in L_R$, the following holds:*

  Let $p(z)$ be the probability that V accepts on input z in interaction with P^*. Now if $p(z) > \kappa(z)$, then on input z and access to the oracle P_z^* (P^* with fixed input z), the extractor K outputs a string in $R(z)$ within an expected number of steps bounded by
  $$\frac{|z|^c}{p(z) - \kappa(z)}.$$

By having access to the oracle P_z^*, we mean the possibility to reset it to a previous state, including the state of the random tape.

This definition makes no statement about the case $z \notin L_R$, e.g., it allows P to "convince" V that it knows a non-trivial factor of a prime. This is ok in our application where V will choose $z \in L_R$.

Further note that cheating provers are not restricted to polynomial time here, because the time that P^* needs comes in indirectly via the time each oracle call of K needs. We will say more about this in the context of interactive arguments. This also implies that there would be no need for a computational security parameter; however, one could easily modify the definition so that the tolerated knowledge error (or typically a negative logarithm of it) is a free input parameter, instead of a function of z. The same could then be done for our following definition.

As outlined in the introduction, we now define soundness of an interactive argument for the case where it is based on the prover's presumed inability to solve a fixed instance of a computational problem. We use a relation R as described above to model this computational problem, and the definition describes the idea that the prover can only argue something false by instead demonstrating knowledge of a solution to the given problem instance from R.

In order to get one uniform extractor, and not a different one for each value x, we include the values x in the relation.

Definition 4.2 Relative definition of interactive arguments
Let L be a language, R a binary relation, $\sigma : \{0,1\}^* \to [0,1]$, and (P, V) a pair of probabilistic polynomial-time interactive Turing machines, taking two common inputs x and z. We say that (P, V) is an interactive argument for L with soundness error σ relative to R *if it has the following properties:*

- *Completeness: If $x \in L$ and $z \in L_R$, and P is given an appropriate auxiliary input (depending on x), the probability that V rejects on input (x, z) when interacting with P is negligible in the minimum of $|x|$ and $|z|$.*
- *Soundness relative to R: We require that V satisfies the validity condition of Definition 4.1, considered as a knowledge verifier for the relation*

$$R' = \{((x, z), y) | x \notin L, (z, y) \in R\}.$$

with knowledge error σ.

Note that we do not need the non-triviality condition of Definition 4.1. In our case it would mean that if the prover knows a solution to the problem instance z, he can in fact cheat the verifier.

The soundness condition, written out, means that for every prover P^*, every $x \notin L$, and every $z \in L_R$, the following holds: Let $p(x, z)$ be the probability that V accepts in interaction with P^* on input (x, z). If $p(x, z) > \sigma(x, z)$ (for simplicity we omitted an additional pair of parantheses denoting that σ operates on the pair as one string), then the knowledge extractor K, on input (x, z) and access to the oracle $P^*_{x,z}$, outputs a string in $R(z)$ within an expected number of steps bounded by

$$\frac{(|x| + |z|)^c}{p(x, z) - \sigma(x, z)}.$$

As in the definition of proofs of knowledge, cheating provers are not restricted to polynomial time in our soundness condition. The prover's running time comes in as follows: If a protocol satisfies the soundness condition and some prover P^* with running time $T(x, z)$ can cheat with probability higher than $\sigma(x, z)$, there is an algorithm solving the given instance z in expected time $T(x, z)$ times the number of steps given above.

We now show that this relative definition implies the absolute definition of soundness if the underlying problem is indeed hard.

Proposition 4.3 *Let (P, V) be an interactive argument relative to R with negligible soundness error σ. Let gen be a polynomial-time algorithm that, given a security parameter k, chooses an element $z \in L_R$ such that one assumes that no probabilistic polynomial-time algorithm F, on input (k, z), can find $y \in R(z)$ with more than negligible probability (in k). Moreover, we assume that gen guarantees $|z| \geq k$.*

Let (P_1, V_1) be the protocol with input x where the verifier first chooses z using $gen(|x|)$, and then (P, V) is run on input (x, z). Then (P_1, V_1) is an interactive argument for L according to the fully uniform Definition 3.2.

Proof It is clear that P_1 and V_1 are polynomial-time in $|x|$, and completeness follows with the additional condition $|z| \geq k = |x|$.

Soundness: Assume there is a pair (M_1^*, P_1^*) of a probabilistic polynomial-time message finder and main prover contradicting Definition 3.2. Let their joint concrete running time on input k be $T_1(k)$ and their success probability $p_1(k)$. The assumption that this pair contradicts negligible soundness means that there is a constant c_1 such that $p_1(k) > k^{-c_1}$ for infinitely many values k.

Let K be the knowledge extractor and c the constant guaranteed by the relative Definition 4.2 for (P, V).

Now we construct an algorithm F that breaks the assumption about the underlying relation R: On input (k, z), the algorithm F first calls the message finder M_1^* with input k to obtain a value x and some $view_{M_1^*}$. If the length of x is not k, then F aborts. Otherwise it calls the knowledge extractor K with the input (x, z) and answers its oracle questions by calling $P_1^*(x, view_{M_1^*})$ and giving P_1^* the value z in the first step. Note that this oracle $P^*(x, z)$ is a valid oracle for K. Let its success probability in cheating V be $p(x, z)$. We have $p_1(k) = \sum p_{M_1^*,k}(x) p_{gen,k}(z) p(x, z)$, where $p_{M_1^*,k}(x)$ and $p_{gen,k}(z)$ are the probabilities with which $M_1^*(k)$ and $gen(k)$ output x and z, respectively.

F lets K run for

$$2\frac{(k + |z|)^c}{p_1(k)/2 - \sigma(x, z)}$$

steps if

$$p_1(k) > k^{-c_1} > 4\sigma(x, z); \qquad (*)$$

otherwise it stops at once. As σ is negligible and the length of the pair (x, z) is at least k, there are still infinitely many values k where $(*)$ holds for all (x, z). We have to show that F is polynomial-time and its success probability non-negligible.

Size: Each step is at most a call to $M_1^*(k)$ or P_1^* on an input produced by $M_1^*(k)$, and either runs for at most $T_1(k)$ steps. The factor $(k + |z|)^c$ is polynomial because z is chosen by $gen(k)$. Finally, if any steps are carried out, the denominator gives at most a factor of $4k^{c_1}$.

Probability: It is sufficient to consider the values k for which $(*)$ holds. By Definition 4.2, K outputs a string in $R(z)$ in an expected number $(k+|z|)^c/(p(x, z) - \sigma(x, z))$ of steps whenever $x \notin L$ and $p(x, z) > \sigma(x, z)$. By Markov's rule, it makes such an output with probability at least 1/2 if run for twice this number of steps. All conditions are fulfilled if $x \notin L$ and $p(x, z) > p_1(k)/2$. By Markov's rule, this is the case with probability at least $p_1(k)/2$ (over the independent choices of x and z; recall that $p_1(k)$ is the weighted average of the values $p(x, z)$).

Altogether, F therefore has a success probability of at least $k^{-c_1}/4$ if its input z is chosen by $gen(k)$ for an infinite number of values k. This contradicts the assumption about the underlying relation R.

□

5 Iteration in the Fixed-Instance Case

Bellare and Goldreich show in [4] that the knowledge error for a protocol iterated sequentially decreases almost exponentially with the number of iterations:

Theorem 5.1 *Let V be a knowledge verifier for relation R with soundness error κ. Assume that an element $y \in R(z)$ can be found in time $2^{l(z)}$ for any $z \in L_R$, where l is at most polynomial. Then the interactive algorithm V^m consisting of $m(z)$ independent sequential iterations of V on the same input z is a knowledge verifier for relation R with knowledge error $(1 + 1/l)\kappa^m$. (Recall that l, κ, and m are all functions of z.)*

By the relative definition of interactive arguments, this immediately implies that the soundness error of the m-fold iteration of such an argument on the same input (x, z), i.e., with a fixed instances, decreases in exactly the same way.

In [4], the question was raised of whether the factor $(1 + 1/l)$ can be removed. We show in the following that this is possible in an important class of special cases. More importantly, we also provide a tighter reduction. The special cases are defined as follows:

Definition 5.2 (Sharp threshold extractor) *Let K be a knowledge extractor for knowledge verifier V and relation R. The machine K is called a* sharp threshold extractor *if the following is satisfied: for any prover P^* that on input z convinces V with probability larger than $\kappa(z)$, K using P^* as oracle runs in an expected number $f(|z|)$ of steps for some fixed polynomial f.*

Many, if not all known proofs of knowledge that one wants to iterate in practice have sharp threshold extractors. As an example, consider a protocol consisting of the verifier asking the prover one out of a polynomial number t of questions, and where the knowledge can be computed in polynomial time from correct answers to more than $g < t$ questions. Such a protocol has knowledge error g/t and a sharp threshold extractor, because any prover who convinces the verifier with probability greater than g/t must be able to answer at least $g + 1$ questions, all of which can be found in a polynomial number of steps by rewinding the prover (at most t times).

Theorem 5.3 *Let V be a knowledge verifier for relation R with knowledge error κ. Assume that there is a sharp threshold extractor for V. Then the interactive algorithm V^m consisting of $m(z)$ independent sequential iterations of V on the same input z is a knowledge verifier for relation R with knowledge error κ^m.*

Proof We fix an input z, so let $m(z) = m$ and $\kappa(z) = \kappa$. Let a be the maximum number of random bits consumed by V during one iteration and $t = 2^a$. Clearly, t is the maximal number of distinct interactions that could take place between V and any fixed prover (including a fixed random tape). Let g be the maximal integer with $g/t \leq \kappa$.

Fix an arbitrary prover P^{m*} convincing V^m on input z with probability $p > \kappa^m$. Let $p(r)$ be the probability that P^{m*} with a specific random tape r convinces V^m.

We consider the proof tree for V^m, P^{m*} with a fixed random tape r, and z. In this case, the edges out of a node correspond to the t possible values of V's random choices in that execution, but those for which V rejects are deleted. A level i is the set of nodes at a fixed distance i from the root. The nodes in level m, which correspond to final acceptance by V, are called endnodes and their number is called *end*. In our case,

$$end = p(r)t^m. \qquad (1)$$

A node is said to be *good* (for the extractor) if it has more than g children. Let $Good_i$ be the number of good nodes in level i. We claim that for all trees, the number of endnodes is

$$end \leq g^m + \sum_{i=0}^{m-1} (t-g)g^{m-i-1}Good_i. \qquad (2)$$

We show (2) by induction on the number of levels with non-zero value of $Good_i$: If $(Good_0, ..., Good_{m-1}) = (0, ..., 0)$, the maximal number of endnodes occurs if all internal nodes have g children, i.e., we get g^m endnodes. Next, assume the formula holds for any tree with good nodes only up to level $j-1$. Consider a tree with a sequence of the form $(Good_0, ..., Good_j, 0, ..., 0)$. Make a new tree by removing enough children (and all their successors) from the good nodes at level j so that they have g children left. For each node, we have removed at most $t-g$ children on level $j+1$, and each of these can lead to at most $g^{m-(j+1)}$ endnodes, as all nodes are bad from level $j+1$ onwards. The induction hypothesis applies to the remaining tree, and adding to this the number of endnodes we removed gives the desired result.

The overall strategy for the knowledge extractor will be to find a good node as quickly as possible, basically by trying random nodes of random trees. The above claim gives us a lower bound on the number of good nodes, except that the summation is weighted according to the level of the node. We will therefore not try levels uniformly, but choose each level i with a probability p_i carefully adapted to make the most of the weighted sum: For $i = 0, ..., m-1$, let

$$p_i = t^i g^{m-i-1} p_{min}, \quad \text{where} \;\; p_{min} = \frac{t-g}{g^m((t/g)^m - 1)}.$$

It is easy to verify that these probabilities add up to 1. Now consider the following algorithm for a knowledge extractor for V^m:

Repeat the following loop until an element in $R(z)$ has been found:

1. Choose the random tape r of P^{m*} uniformly, and choose a level i, using the probability distribution $p_0, ..., p_{m-1}$.
2. Try to select a node in level i by simulating the protocol, i.e., running the algorithm V^m (with new random choices each time) and using the oracle P_z^{m*} with the random tape r. If V^m rejects before we reach level i, go back to step 1.

   (Note that this means selecting one of the t^i potential nodes in level i with uniform probability; it is reached if and only if it is in fact in the tree.)
3. Run the sharp threshold extractor K that we have for V for this node, hoping that it is a good node. This means that we answer K's oracle queries by rewinding P_z^{m*} to the situation after step 2 each time. If K outputs an element in $R(z)$ within $2f(|z|)$ steps, where f is the polynomial guaranteed by Definition 5.2, we output this element and stop. Else, go to step 1.

Let $p^*(r)$ be the probability that we reach step 3 with a good node for a specific r. We can bound $p^*(r)$ using formulas (1) and (2) and the definition of the p_i's:

$$\begin{aligned}
p^*(r) &= \sum_{i=0}^{m-1} p_i Good_i / t^i \\
&= p_{min} \sum_{i=0}^{m-1} g^{m-i-1} Good_i \\
&= \frac{1}{g^m((t/g)^m - 1)} \sum_{i=0}^{m-1} (t-g) g^{m-i-1} Good_i \\
&\geq \frac{1}{g^m((t/g)^m - 1)} (p(r) t^m - g^m) \\
&= \frac{p(r) - (g/t)^m}{1 - (g/t)^m} \\
&\geq p(r) - (g/t)^m \\
&\geq p(r) - \kappa^m.
\end{aligned}$$

The overall probability p^* that we reach step 3 with a good node is therefore at least $p - \kappa^m$.

If we reach step 3 with a good node, K succeeds in expected number of steps $f(|z|)$, because running P_z^{m*} with a fixed starting state for one iteration is a valid oracle for K and V accepts in good nodes with probability at least $(g+1)/t \geq \kappa$. The probability that K runs for more than twice its expected number of steps is at most $1/2$ by Markov's rule. Hence the expected number of times we restart the loop from step 1 is at most $2/(p-\kappa^m)$, and the number of steps each time is clearly polynomial. □

Corollary 5.4 *Let (P, V) be an interactive argument for language L with soundness error σ relative to R. Assume that there is a sharp threshold extractor for V, considered as knowledge verifier for the relation R' defined as in Definition 4.2. Then the protocol (P^m, V^m) consisting of $m(z)$ independent sequential iterations of (P, V) on the same input (x, z) is an interactive argument for language L with soundness error σ^m relative to R.*

Proof The completeness condition can easily be seen, and the soundness condition carries over immediately from the theorem. □

Let us elaborate a little on what this result means in practice. It does not allow us to say that the prover *cannot* cheat with probability better than σ^m. But we can say how many computing resources he would need to reach any given goal above σ^m. If we look closely at the reduction in the above proof, we see that doing the loop once corresponds to emulating at most one run of all m iterations, plus running the knowledge extractor for at most $2f(|z|)$ steps. Hence we get:

Corollary 5.5 *With the notation and assumption as in Corollary 5.4: Any prover P^* that makes the verifier accept all m iterations with probability $p(z) > \sigma(z)^m$ can be converted into an algorithm that finds an element in $R(z)$ in expected time*

$$2\frac{T_{P^*}(z)(1+2f(|z|)) + mT_V(z)}{p(z)-\sigma(z)^m},$$

where $T_{P^*}(\cdot)$ and $T_V(\cdot)$ are the running times of P^* and V, respectively.

Of course, the term $f(|z|)$ coming from the knowledge extractor also includes oracle calls. For most known protocols, we can evaluate this further:

Corollary 5.6 *If we further specialize Corollary 5.5 to the very common case of protocols where one out of t questions (t constant) is asked in each round, the expected running time is bounded by*

$$2\frac{T_{P^*}(z)(1+t) + mT_V(z)}{p(z)-\sigma(z)^m}.$$

Note that the expression in the denominator of this corollary is linear in $T_{P^*}(z)$, and therefore gives a very tight connection between the time needed to cheat V and the time needed to break the computational assumption.

6 Independent Iterations in the Uniform Model

For completeness, we now briefly consider the case we have not yet looked at, namely independent iterations in the uniform model.

Proposition 6.1 *Let (P, V) be an interactive argument for a language L according to Definition 3.2 with constant soundness error $c < 1$.*

Then (P^m, V^m), the protocol consisting of $m(|x|) = |x|$ independent sequential iterations of (P, V) on the same input x, is an interactive argument for L in the sense of Definition 3.2 with negligible soundess error.

Proof sketch

The proof is very similar to the one for the fixed instance case, and so is only loosely sketched here. Let a message finder M^* and a prover P^{m*} be given that violate the conclusion of the proposition. The same message finder will be used against V. On a certain fraction of its outputs, P^{m*} cheats V^m with significant probability. For each such output, we form the proof tree for P^{m*}, V^m. If every node in the tree is bad, i.e., the ratio of outgoing edges to the maximum possible is less than c, then the accept probability would be at most c^m, i.e., exponentially small. If fact it is much larger, namely a polynomial fraction by assumption, and so there must be a large number of good nodes, one of which can be found using a similar algorithm as in the proof for the fixed instance case. Using such a node, we can build a prover contradicting the assumption on (P, V). □

7 An Efficient Zero-Knowledge Argument for NP

In this section we present an efficient statistical zero-knowledge argument for Boolean circuit satisfiability, and hence for any NP problem, by the NP-completeness of circuit-SAT and standard reductions. The protocol can be based on the existence of collision-intractable hash functions, i.e., easily computable functions that map inputs of (in principle) any length to a fixed length output, and for which it is hard to find different inputs leading to the same output. Our construction combines three ingredients:

- The unconditionally hiding multi-bit commitment scheme by Damgård, Pedersen, and Pfitzmann [8, 9], based on any collision-intractable hash function family H. The receiver of the commitments chooses a hash function $h \in H$ with output length $k + 1$, where k is a security parameter. (The choice is made once and for all, and the functions have short descriptions.) An m-bit string can then be committed to by a commitment of length $k + 1$ bits and opened by sending $10(k+1)$ bits, plus the m bits of the string itself, of course. The scheme guarantees that a commitment to x only reveals an exponentially small (in k) amount of Shannon information about x. Moreover, any method for making a commitment and opening it in two different ways easily leads to a collision for h.

- The BCC protocol [1] for showing that a Boolean circuit is satisfiable. It works based on any bit commitment scheme for single bits and is a computational zero-knowledge proof system or a perfect/statistical zero-knowledge argument, depending on whether the commitments used are computationally or unconditionally hiding. The basic step in the protocol is that the prover commits to $O(m)$ bits, where m is the size of the circuit, and depending on a random challenge from the verifier, the prover either opens all the bits or a specific subset of them that depends on the satisfying assignment. This basic step is iterated a number of times.

- The method by Kilian, Micali, and Ostrovsky from [16] for using a multi-bit commitment scheme in any protocol of a type they call "subset-revealing", of which the BCC protocol is an example. The interesting point is that the method works even if the commitment scheme does not allow opening individual bits in a multi-bit commitment. The method replaces each basic step of the original protocol by a new one which needs 5 messages instead of 3 and contains 2 commitments to $O(m)$ bits each instead of $O(m)$ commitments to 1 bit each. If the prover could cheat in the old basic step with probability at most 1/2, he can cheat in the new one with probability at most 3/4 (without breaking the computational assumption).

Let (P, V) denote the protocol that takes as input a circuit C and a hash function $h \in H$, and executes one basic step obtained by modifying the basic step of BCC by the method of [16] and the commitment scheme of [8, 9] using the given hash function h.

Let R be the relation $\{(h, (y, y')) | \ h \in H, h(y) = h(y'), y \neq y'\}$, i.e., the underlying computational problem of finding collisions. The relation R' used in Definition 4.2 is then

$$\{((C, h), (y, y')) | \ C \text{ non-satisfiable}, (h, (y, y')) \in R\}.$$

It is easy to see that V is a knowledge verifier for R' with knowledge error 3/4 and a sharp threshold extractor: In the protocol, V chooses at random between a total of 4 challenges for the prover, and given satisfactory answers to all of them, one can compute a satisfying assignment for C or break the commitment scheme, i.e., find a collision for h. It follows immediately that (P, V) is an interactive argument for Boolean circuit satisfiability with soundness error 3/4 relative to R. Thus Corollary 5.4 implies that the iterated protocol (P^m, V^m) only has soundness error $(3/4)^m$. Finally, we can apply Proposition 4.3 to the overall protocol (P_1^m, V_1^m). We summarize this protocol and the results:

1. The common input is a circuit C of size m.

2. V_1^m chooses at random a function $h \in H$ with output length m.

3. Run (P, V) on input (C, h) for m iterations. V_1^m accepts if and only if all iterations were accepting.

Theorem 7.1 *Assume that H is a family of collision-intractable hash functions. Then (P_1^m, V_1^m) is a statistical zero-knowledge argument for Boolean circuit satisfiability according to Definition 3.1 with the following properties: The protocol requires communicating $O(m^2)$ bits. The subprotocol (P^m, V^m) has soundness error $(3/4)^m$ relative to R. Concretely, if any probabilistic polynomial-time prover P^* can cheat with probability $\epsilon(m) > (3/4)^m$ in expected time $T(m)$, there is a probabilistic algorithm that finds collisions for the hash function used in expected time dominated by the term $10T(m)/(\epsilon(m) - (3/4)^m)$.*

As briefly mentioned near the definitions, for a protocol of the type we consider, there are actually a number of parameters, which one may set *independently* in practice: the size of the input circuit, say n, the logarithm of the probability with which we allow the prover to cheat (assuming he cannot break the hash function), which corresponds to m, and the output length k of the hash function. To simplify, we have let all parameters be $O(m)$.

Using the BCC protocol based on a 1-bit commitment scheme would give a communication complexity of $O(m^3)$ bits. Kilian [14, 15] has found a protocol based on probabilistically checkable proofs that would, with our choice of parameters, have a communication complexity of $O(m^2 \log m)$. However, our protocol would not be superior to Kilian's for all choices of parameters – in fact Kilian shows that the communication complexity does not have to depend on the size of circuit at all. Using a completely different method, Cramer and Damgård [7] obtained an argument that also has $O(m^2)$ complexity. In comparison, their protocol is perfect zero-knowledge and constant-round, and if one considers the parameters independently, [7] has asymptotically smaller communication complexity for small soundness error values. On the other hand, our protocol is computationally more efficient for the concrete constructions known.

Perhaps even more interesting is the performance in practice. For instance, if we use SHA-1 as the hash function, which has a 160-bit output, and we set the soundness error at 2^{-50}, then a circuit consisting of 10000 gates could be proved satisfiable using about 3 Mbyte of communication.

To assess the computation effort required, it seems reasonable to assume that an implementation would spend almost all its time hashing. SHA-1 can be implemented on standard PC's at speeds around 6-8 Mbyte/sec. This suggests that, at a security level of 2^{-50}, a real implementation should be able to handle around 20000 gates per second, assuming that the communication lines can keep up. For instance, a circuit part for proving that a secret value is a DES key encrypting a certain plaintext block into a certain ciphertext block can be handled in less than 2 seconds. (The main part of the circuit are 16 times 8 S-boxes. Each S-box is a table with 2^6 rows of 4 fixed entries, which can easily be implemented with about 2 gates per row for addressing and one gate per entry "1".)

To the best of our knowledge this is the most practical protocol proposed for circuit satisfiability.

References

[1] G. Brassard, D. Chaum, and C. Crépeau, "Minimum Disclosure Proofs of Knowledge," *J. Computer and System Sciences*, vol. 37, pp. 156-189, 1988.

[2] G. Brassard, C. Crépeau, S. Laplante, and C. Léger, "Computationally Convincing Proofs of Knowledge," *Proc. 8th Annual Symp. Theoretical Aspects of Computer Science (STACS 91)*, Berlin: Spinger Verlag, 1991, pp. 251-262.

[3] G. Brassard, C. Crépeau, M. Yung, "Constant-round perfect zero-knowledge computationally convincing protocols," *Theoretical Computer Science*, vol. 84, no. 1, pp. 23-52, 1991.

[4] M. Bellare and O. Goldreich, "On Defining Proofs of Knowledge", in *Advances in Cryptology - Proc. CRYPTO '92*, Berlin: Springer-Verlag, 1993, pp. 390-420.

[5] L. Babai and S. Moran, "Arthur-Merlin Games: A Randomized Proof System and a Hierachy of Complexity Classes", in *JCSS*, vol.36, 254-276, 1988.

[6] M. Bellare, R Impagliazzo, and M. Naor, "Does Parallel Repetition Lower the Error in Computationally Sound Protocols?", *in Proc. 38th IEEE Symp. Foundations of Computer Science*, 1997.

[7] R. Cramer and I. B. Damgård, "Linear Zero-Knowledge - A Note on Efficient Zero-Knowledge Proofs and Arguments", in *Proc. 29th Annual ACM Symp. Theory of Computing*, 1977, pp. 436-445.

[8] I. B. Damgård, T. P. Pedersen, and B. Pfitzmann, "On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures," in *Advances in Cryptology - Proc. CRYPTO '93*, Berlin: Springer-Verlag, 1994, pp. 250-265.

[9] I. B. Damgård, T. P. Pedersen, and B. Pfitzmann, *Statistical Secrecy and Multi-Bit Commitments*, BRICS report series RS-96-45, Aarhus University, Dept. of Computer Science, 1996, available at http://www.brics.dk. To appear in IEEE Trans. Inform. Theory, probably May 1998.

[10] L. Fortnow, "The Complexity of Perfect Zero Knowledge," in *Proc. 19th Annual ACM Symp. Theory of Computing*, 1987, pp. 204-209.

[11] O. Goldreich, *Foundations of Cryptography (Fragments of a Book)*, Dept. of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, Feb. 23, 1995, available at ftp.wisdom.weizmann.acil//pub/oded/bookfrag.

[12] S. Goldwasser and S. Micali, "Probabilistic encryption," *J. Computer and System Sciences*, vol. 28, pp. 270-299, 1984.

[13] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Computing*, vol. 18, no. 1, pp. 186-208, 1989.

[14] J. Kilian, "A Note on Efficient Zero-Knowledge Proofs and Arguments," in *Proc. 24th Annual ACM Symp. Theory of Computing*, 1992, pp. 723-732.

[15] J. Kilian, "Efficient Interactive Arguments," in *Advances in Cryptology - Proc. CRYPTO '95*, Berlin: Springer-Verlag, 1995, pp. 311-324.

[16] J. Kilian, S. Micali, and R. Ostrovsky, "Minimum resource zero-knowledge proofs," *in Proc. 30th IEEE Symp. Foundations of Computer Science*, 1989, pp. 474-479.

[17] *Secure Hash Standard*, Federal Information Processing Standards Publication FIPS PUB 180-1, 1995.

Recent BRICS Report Series Publications

RS-97-50 Ivan B. Damgård and Birgit Pfitzmann. *Sequential Iteration of Interactive Arguments and an Efficient Zero-Knowledge Argument for NP*. December 1997. 19 pp.

RS-97-49 Peter D. Mosses. *CASL for ASF+SDF Users*. December 1997. 22 pp. Appears in *ASF+SDF'97, Proceedings of the 2nd International Workshop on the Theory and Practice of Algebraic Specifications, Electronic Workshops in Computing*, http://www.springer.co.uk/ewic/workshops/ASFSDF97. Springer-Verlag, 1997.

RS-97-48 Peter D. Mosses. *CoFI: The Common Framework Initiative for Algebraic Specification and Development*. December 1997. 24 pp. Appears in Bidoit and Dauchet, editors, *Theory and Practice of Software Development. 7th International Joint Conference CAAP/FASE*, TAPSOFT '97 Proceedings, LNCS 1214, 1997, pages 115–137.

RS-97-47 Anders B. Sandholm and Michael I. Schwartzbach. *Distributed Safety Controllers for Web Services*. December 1997. 20 pp. To appear in *European Theory and Practice of Software. 1st Joint Conference FoSSaCS/FASE/ESOP/CC/TACAS*, ETAPS '97 Proceedings, LNCS, 1998.

RS-97-46 Olivier Danvy and Kristoffer H. Rose. *Higher-Order Rewriting and Partial Evaluation*. December 1997. 20 pp. Extended version of paper to appear in *Rewriting Techniques and Applications: 9th International Conference*, RTA '98 Proceedings, LNCS, 1998.

RS-97-45 Uwe Nestmann. *What Is a 'Good' Encoding of Guarded Choice?* December 1997. 28 pp. Revised and slightly extended version of a paper published in *5th International Workshop on Expressiveness in Concurrency*, EXPRESS '97 Proceedings, volume 7 of Electronic Notes in Theoretical Computer Science, Elsevier Science Publishers.

RS-97-44 Gudmund Skovbjerg Frandsen. *On the Density of Normal Bases in Finite Field*. December 1997. 14 pp.

RS-97-43 Vincent Balat and Olivier Danvy. *Strong Normalization by Run-Time Code Generation*. December 1997.